2012.06

PROGRAMMER 程序员



产品经理

18 OpenStack，火热正当时

58 金山云CTO杨钢
情系网盘

75 狂奔的移动端用户体验设计

78 Cocos2D前世今生

83 跨平台游戏开发利器

91 Siri详解
执行和服务系统

96 可伸缩性的10年探索

107 个性化推荐技术十大挑战

117 学院派的复兴
PostgreSQL 和 MySQL 应用场景对比

122 Ready? Go!
Go语言并发模型和工具链

126 稳定婚姻问题
Gale-Shapley算法

136 硅谷新传奇Kevin Systrom

ISSN 1672-3252
9 771672 325128 06

邮发代号：2-665　定价：15元

www.programmer.com.cn　www.csdn.net

Contents





P6

资讯

6 新闻

8 新产品新工具

10 外刊速递

14 程序天下事

18 OpenStack，火热正当时
——OpenStack Design Summit & Conference 2012速递

20 旁观ACM ICPC

22 选择标准的、代表行业趋势的平台
——微软平台战略部总经理Tim O'Brien专访



P26

封面报道：产品经理

本期封面报道，分享了多位一线产品经理的成长心路和实战经验，涵盖优秀产品经理所应具备的素质和技能、产品经理该如何培养逻辑思维、如何将用户变成布道者、如何更好地解决与技术人员之间的意见分歧、如何进行更有效的市场调研等多个热门话题，更有产品经理常犯的错误和如何警惕恶性循环等来自实践的反思和总结……

27 优秀产品经理必备的素质和技能

30 产品经理的逻辑思维

33 如何将用户变成布道者

36 产品经理与项目经理的关系

38 做不了，谁的错

40 如何寻找优秀的产品经理

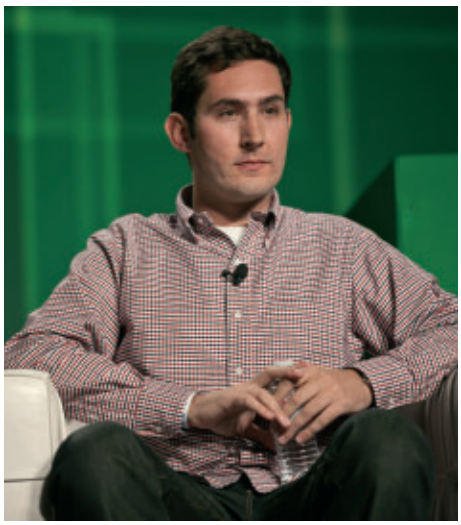
43 技术人员如何转行做产品经理

46 产品经理在市场调研中扮演的角色

49 数据启示的互联网产品设计

52 产品经理常犯的七大错误

55 警惕恶性循环



P136



P134

管理

58 情系网盘
——专访金山云CTO杨钢

60 时间的旅行者
——计划与估算导论

64 敏捷中的控制，控制中的敏捷

移动

70 马拉松前100米的风景
——《智能360》创始人李传丰专访

72 声效表达在游戏沉浸方面的影响力

75 狂奔的移动端用户体验设计（下）
移动终端面临着复杂的环境，作者以实例为基础，详细从可控感、速度感、好感化三大方向，阐述了如何抓住用户体验设计的重心。

78 Cocos2D家族的前世今生
Cocos2D,这款用于开发2D游戏的开发框架以及由其衍生的Cocos2D—X跨平台开发框架和众多分支，正在帮助越来越多的开发者实现高效稳定的游戏开发。《程序员》近日专访了Cocos2D—iPhone作者Ricardo Quesada，请他分享Cocos2D家族的发展历程与未来方向。

81 Cocos2D-X框架中间层演进

83 跨平台游戏开发利器
——Cocos2D—HTML5开源2D游戏引擎介绍
Cocos2D—HTML5是基于HTML5规范集的Cocos2D引擎的分支，于2012年5月发布。Cocos2D—HTML5的作者林顺将在本文中介绍Cocos2D—HTML5的框架、API、跨平台能力以及其强大的性能。

87 Cocos2D-X for XNA游戏开发指南（上）
本文介绍了XNA版本相对于C++版本中CCApplication、CCNode的变化，绘制、管道、资源等注意事项以及对集合和数组的处理。下期文章将通过一个开源的Cocos2D游戏展示简单游戏的开发过程。

云计算

91 Siri详解之执行和服务系统
在4月刊《程序员》刊登的《Siri详解之输入系统和活跃本体》一文中，作者从技术层面详解Siri，主要讲述了输入系统和活跃本体两部分内容。本文将继续深入Siri的技术层面，阐释执行系统、服务系统和输出系统三部分内容。

96 可伸缩性的10年探索
——知名网站的技术实现
网站需要具有良好的可伸缩性，来应对不断增长的访问量和数据量。本文结合Google、Facebook等知名网站的技术发展历程，阐述了它们在可伸缩性上的技术实现。

2012年06月刊　总第236期

Contents

主管：中国社会科学院
主办：中国社会科学院文献信息中心
出版：《程序员》杂志社
网址：http://www.programmer.com.cn
国际刊号：ISSN 1672-3252
国内刊号：CN11-5038/G2
邮发代号：2-665
广告经营许可证号：京东工商广字0188号

总编：黄长著　Editor-in-chief：Huang Changzhu
社长/常务副总编：张悦校　President：Zhang Yuexiao
副社长：蒋 涛　Vice President：Jiang Tao
编委会：黄长著　张悦校　陈洋彬　蒋涛　曾登高　刘江
Editorial Member：Huang Changzhu　Zhang Yuexiao　Chen Yangbin
Jiang Tao　Zeng Denggao　Liu Jiang

执行总编：刘江　Executive Editor-in-chief：Liu Jiang
执行主编：孟迎霞　Managing Editor：Meng Yingxia
编辑部主任：董世晓　Director：Dong Shixiao
责任编辑：陈博　杨爽　卢鹃翔
Editors：Chen Bo　Yang Shuang　Lu Dongxiang
特邀编辑：蔡学镛　池建强　冯大辉　高博　肖新光　杨栋　余晟　周爱民
Contributing Editors：Cai Xueyong　Chi Jianqiang　Feng Dahui　Gao Bo
Xiao Xinguang　Yang Dong　Yu Sheng　Zhou Aimin
美术设计：纪明超　Art Designer：Ji Mingchao
美术编辑：朱凯　Art Editor：Zhu Kai
流程编辑：白羽中　Coordinator：Bai Yuzhong
Tel：010-64351458
E-mail：editor@csdn.net

广告总代理：北京创新乐知信息技术有限公司
Sole Advertising Agency：Bejing CSDN Co.,Ltd
Tel：010-64376055
E-mail：ad@csdn.net
Marketing Dept：010-51661202（ext 149）
E-mail：market@csdn.net

发行部
Distribution Dept.010-64351431
E-mail：sales@csdn.net

读者服务部
Readers service Dept.
网上订购：http://dingyue.programmer.com.cn/
读者信箱：reader@csdn.net
地址：北京市朝阳区广顺北大街33号院1号楼福码大厦B座12层
Address: B-12th Floor Fairmont Tower NO.33 Guangshun North street,Chaoyang District,Beijing
邮政编码：100102
电话：010-64351436
传真：010-64348545

法律顾问：北京中润律师事务所 王杰
Law Consultant：Beijng Zhongrun Lawyer Firm
印刷：北京盛通印刷股份有限公司
Print: Beijing Shengtong Printing Co., Ltd.
出版日期：每月1日
Publication Date：the first day per month
零售价：RMB 15.00元　新台币 390元　HK $ 35.00（港、澳）
US $ 9.00（海外）
Retail Price：RMB 15, NT$390, HK $ 35.00,US $ 9.00

本刊文章版权所有 未经许可不得转载
发现装订错误或缺页，请将杂志寄回本刊读者服务部调换

100 云计算与B2B2C架构实战
102 基于P2P网络的分布式消息队列
107 个性化推荐技术的十大挑战

技术

112 ThoughtWorks技术雷达（2012年3月）
自2011年7月《技术雷达》发布后，以下几个趋势越来越明显：SQL数据存储替代产品的持续发展，谨慎对待从UI到测试的所有代码，基于浏览器的语言和技术更加严谨与多样性化，应用程序和服务更小、更简单、更快速。

117 学院派的复兴
——PostgreSQL和MySQL应用场景对比
近年来，由于开源社区的努力，以Hadoop、MangoDB为首的NoSQL技术迅速为国人所青睐，而传统关系型数据库领域则喜忧参半。在本文中，作者将结合自己的体会，分析最适合PostgreSQL的应用场景。

122 Ready? Go!
——Go语言并发模型和工具链
在《程序员》5月刊中，作者介绍了Go语言的部分语法和类型系统，本文作为延续，将重点介绍Go语言的并发模型和工具链。

126 稳定婚姻问题和Gale-Shapley算法
什么是算法？每当有人问作者这样的问题时，他总会引用这个例子：假如你是一个媒人，有若干个单身男子登门求助，还有同样多的单身女子也前来征婚。如果你已经知道这些女孩儿在每个男孩儿心目中的排名，以及男孩儿们在每个女孩儿心中的排名，你应该怎样为他们牵线配对呢？

129 1024核CPU上的并行编程
摩尔定律在多核时代的表现形式是，内核数量几乎也会以成倍的速度增加，但内核数量的增长是否能与性能的增强划上等号呢？本文将探讨在这种背景下，如何编写程序才更有效。

百味

132 新书上架
134 Geek
136 硅谷新传奇Kevin Systrom

企业专栏

24 产品新人的算术课
——需求规划能力修炼
56 走进MPD，180分钟改变研发团队

移动

2012年5月，云笔记软件服务商Evernote宣布进入中国市场，成为国内移动互联网领域关注的热点。在此之前，包括Flipboard、Zynga、Popcamp等企业和产品也都在着力加速国内业务的扩展。比如Flipboard结盟新浪和人人；《水果忍者》采取广告运营由多盟代理，并和乐逗游戏、杭州思凯合作；而国内移动广告平台帷千动媒则接管了《愤怒的小鸟》的广告运营。

国外精品应用加快入华步伐，自然也会波及国内开发者。国内移动应用市场的竞争将更加激烈，如果难以和国外同类产品抗衡，就会面临原有用户被逐步分流。资本趋冷和国内互联网巨头的加速布局也进一步挤压了中小开发者的发展空间。



另一方面，传统互联网巨头进入国内的历程也并非一帆风顺，Google、Myspace都有过水土不服的经历。Evernote化身“印象笔记”之后便引发了用户对于安全和隐私的担忧。而应用能否真正跨越国界取得成功，往往也会经受文化等因素的考验，比如日本已出版过几十本关于Evernote的书籍，但国内显然还没有达到这样的接受程度。

知识搜索

早在4月就有用户发现Google搜索能够以类似Wolfram|Alpha的图形化方式显示数学表达式。5月17日，Google发布了Knowledge Graph（知识图谱）功能，这种搜索方式不仅提供与用户搜索词汇匹配的页面，在用户搜索一个能触发Knowledge Graph的关键词时，搜索结果页面的右侧将会出现一个信息框，包含与该关键词具有特别相关性的各种信息。Google搜索工程师Ben Gomes表示：“我们的目标是帮助用户更深地挖掘一个主题，提供一个用于组织信息的框架，这是Google从‘数据到知识’计划的一部分。”如果你使用过Wolfram|Alpha，一定会发现这些展现形式似曾相识。

自从计算机诞生之初，人们就希望赋予计算机直接回答问题的能力。2009年，Stephen Wolfram曾解释过知识搜索的与众不同之处——它并不是从Web页面中寻找相关信息，而是基于系统内大量特定结构数据，通过算法将使用者想要了解的信息或者答案直接“计算”出来。如今，这一目标已离用户越来越近，2009年的Wolfram|Alpha只能理解60%的输入，而如今，它已能对超过90%的输入给出恰当的答案。

越来越多的搜索引擎（比如DuckDuckGo）已不满足于只将与关键词联系最大的链接提供给使用者，互联网搜索正在我们身边潜移默化地发生着转变。

专利

2010年初，Oracle以74亿美元收购了Sun公司，成为Java的所有者。在收购不久后，Oracle指控Google在“开发Android的过程中明知故犯，直接、反复地侵犯了Oracle拥有的Java相关技术的知识产权”，并称Google没有Java API授权许可。

目前法庭裁决Android中的9行代码侵权。乍看起来区区9行代码，相比Android整个系统代码，根本可以忽略不计。但本次审判之所以引人注目，一方面是因为Oracle索赔的10亿美元赔偿金，另一方面是这次被指控侵权的不是Java语言本身，而是其公开API。

电子前哨基金会敏锐地察觉到：“如果Oracle胜诉，意味着API也可以申请版权了。有人敢站出来说自己在开发过程中能一个API也不调用吗？因此，API申请版权对世界的影响显而易见。”

对于上述信息，Google一直表示，API不应该受到版权保护，因为它们更接近程序员所使用构建软件的工具和技术。而Oracle却持相反观点，表示应该采取版权保护，因为它们相比商业秘密法律或是专利更精细和具有针对性。

“JSON是世界上最受欢迎的数据交换格式，我在2001年发现了这种格式，我不能声称说是发明了它，因为它已经自然存在了。”

——Douglas Crockford在谈到JSON时谦逊地说。除了CEO风波，雅虎5月还失去了这位JavaScript技术社区的标志性人物。目前他已经在PayPal担任高级JavaScript架构师。

“个人对用户体验的目标是，做到‘自然’。”

——从Foxmail到QQ邮箱，再到微信，作为国内优秀产品经理代表，张小龙的产品理念颇受关注。日前张小龙在知乎网发表长文分享了自己对用户体验的看法。

“公司将不再接受新用户，此前注册的老用户在6月15日后也将不能使用该服务。”

——Facebook在收购Instagram之后，又再次收购照片分享应用Lightbox。分析人士指出，相对于PC端，它在移动领域进展缓慢，此次收购表明了Facebook将加大移动方面统治的决心。

“我们首次将GPU虚拟化，一个GPU现在可以被无限用户使用。”

——NVIDIA CEO黄仁勋表示，希望公司的显卡被数据中心使用，并用来提高智能手机、平板的显示质量。这是NVIDIA试图将PC显卡业务多样化的新战略。

“我依然是那个喜欢独处的、安静的家伙，我的全部热情就是写出很棒的游戏。”

——Andreas Illiger开发Tiny Wings的想法源于他从小就有一个飞翔之梦。他喜欢制作航模飞机，但由于恐高却至今没有坐过飞机。现在有太多有关暴力和杀戮的游戏，而他想做一个能够打动人心的、轻松的作品。

“演示文稿是专业人士获取及分享知识的重要途径，也能够帮助他们发现新关系并获得能够促进事业成功的新视野。这些恰好是SlideShare能够提供给LinkedIn的。”

——LinkedIn准备以1.19亿美元的价格收购在线文档共享服务提供商SlideShare，在谈到SlideShare的作用时，LinkedIn CEO Jeff Weiner如是说。

“Facebook并不是一支值得长期持有的股票，Facebook何时能真正盈利存疑，其用户年龄偏低，广告效果也有限。”

——纽交所资深交易员Bob Heller在谈到Facebook上市首日股价表现并不如预期理想时作了这番表述。受此影响，当日社交概念股股价集体下跌。

CSDN十大资讯

2012年4月26日–5月26日

01 5月编程语言排行榜

作者 / 夏梦竹

TIOBE榜单前三甲依然为C、Java和C++。在过去的8年中，除了Objective-C挤进前10和Delphi的退出，排行榜Top 10变化很小，开发者采用一门新语言似乎比想象中要困难得多。转变一种编程习惯本身已经不易，将一个大代码库从一门语言迁移至另一门语言的代价更高。

02 Bing进行大规模重新设计

作者 / 张勇

微软宣布将对搜索服务Bing进行重新设计，除了引进包含Facebook、Twitter信息的侧边栏，还将整合Quora等问答网站的相关结果，在帮助用户搜索相关资料的同时，也能引导用户并获得来自各地网络专家的建议。

03 Rovio两高层离职创建新游戏工作室

作者 / 张勇

参与开发《愤怒的小鸟》的两位Rovio领导人——首席服务器架构师Antti Stén和首席美术师Tuomas Erikoinen已离开公司，创建了名为BoomLagoon的游戏工作室。他们的目标是开发一款既能在PC又能在移动平台上运行的游戏，因此他们将使用新的开发语言——HTML5。

04 浏览器巅峰对决

作者 / Main

Pingdom网站发表文章《Current Status of the Browser Wars》，文中引用研究报告数据，对比了全球各个地区的浏览器使用状况。Chrome上升势头不减，Firefox不温不火，而IE则面临着不小的难题，它已流失了欧洲、亚洲、非洲和南美洲的领先优势。

05 雷军与马化腾现场“过招”

作者 / 何清颖

2012全球移动互联网大会上，马化腾和雷军现场对话他们的产品成功经验。马化腾表示因为移动用户试用的时间很短，一个应用，如果在10秒内抓不住用户，就会迅速被删。雷军的观点是，互联网忌讳布局，而强调单点切入，一点点长大的业务才能生存。

06 微信一亿用户背后的架构

作者 / 付江

微信技术总监周颢认为用户会因为系统服务宕机而不断重试，这意味着发生雪崩时，系统可能会承载原先3~10倍的流量，使情况更加恶化。微信有很多“放雪”功能的设计。此外，他认为在任何系统中不要追求完美设计，追求完美设计的是团队往往不能胜任海量服务。



07 为何Xbox成功，Zune惨败

作者 / Robbie Bach

微软前娱乐与设备部门总裁Robbie Bach领导他的团队实现了Xbox的畅销。然而，他领导同一个部门推出的Zune音乐播放器，却没那么成功。Robbie对这两款产品的成败得失进行了总结，分享了他认为创业公司应该学习的四堂课。

08 创业者浪费精力的八个错误

作者 / Jeff Miller

Jeff Miller是美食网页应用Punchfork的创始人，同时也是DuckDuckGo、Ginzametrics、Art.sy、DataMinr及Forkly的投资人。作者通过对自己创业初期所犯的错误选择进行盘点，告诉读者在创业初期应该学会选择，因为在这个阶段，时间是最宝贵的财富。

09 为什么传媒出版巨头都痛恨App

作者 / Jason Pontin

自苹果2010年4月发布iPad起，虽然移动设备让用户从传统的互联网浏览模式中得到了解放，但美国各大出版巨头却纷纷表示自身利益受到了巨大冲击。Nielsen的最新数据显示，4月智能手机与平板电脑用户中，有33%下载了新闻阅读应用，但他们中仅有19%是付费用户。

10 Mozilla控诉微软移动系统垄断

作者 / 杨鹏飞

就在微软即将推出的基于ARM架构的Windows 8之际，Mozilla在博客中发表文章称，他们仿佛又看到了当年IE独霸的黑暗年代。ARM虽然目前主要用于移动设备，但未来很可能也会登陆桌面平台，倘若如此，Mozilla认为历史可能会重演。

蒂姆·库克对苹果的10个重要决定

虽然蒂姆·库克就职苹果CEO时间不久，但他为苹果的发展尽职尽责。美国《eWeek》周刊总结了库克上任后对苹果来说最重要的10个决定。

01 苹果依然是苹果

当库克就任CEO后，他首先明确的就是不会改变公司的运营方式。对他来说，苹果还是那个苹果。股价正在有条不紊地上升，其产品也一如既往吸引着全球各地消费者。

02 不要试图成为史蒂夫·乔布斯

无论人们喜欢与否，都不可否认乔布斯是位奇才，他在技术领域取得了巨大的成就。库克不指望能成为乔布斯那样的标志性人物，尝试模仿不但没有意义，反而会因此犯错。

03 不独占舞台

乔布斯有件事情做得不多，就是与他人分享舞台。每当苹果研发出新产品，大多只由他一个人展示。而库克选择了一条不同的路，他把登上舞台的机会让给公司其他高管，让他们向全世界展示自己对公司所做的贡献。

04 革命性升级并非必要

2011年10月，苹果推出iPhone 4S，几乎所有人都能发现，它并未包含革命性的升级，2012年4月上市的新iPad也同样如此。从这几件事可以看出，库克认为每次产品更新，没必要推翻以往的设计。

05 不忘基层工人

多年来，负责组装苹果产品的富士康员工，早已提出工作环境恶劣、工资不平等问题。库克在2012年开展的公平劳动协会上，亲自审核苹果供应商，并到生产第一线视察，着力解决这些问题。

06 相同的保密文化

乔布斯在苹果创建了保密文化，员工从不会透露新产品的任何信息，以免被解雇。尽管消费者们都希望能从苹果计划中得到些许提示，但库克保持着同样的作风。

07 更安静、友好的苹果

当乔布斯带领苹果时，给人们留下一个好战的印象，他曾扬言要对Android发动“热核战争”。近期库克在财报电话中表示，他想放弃苹果与Android厂商之间的诉讼，向大众展示出苹果安静、友好的一面。

08 运作现金

当乔布斯就职苹果CEO时，巨额现金都堆积在库房。而库克却不同，近期他宣布计划购回公司股份及发行股息，并将部分现金捐献给慈善机构。

09 战略性收购

苹果的战略性收购项目相对较少，库克也许会改变这种情况。苹果近期收购了以色列闪存公司Anobit，根据库克的说法，他认为在正确时间，苹果可以收购更多公司。

10 对中国的承诺

库克很清楚地知道，中国对苹果未来发展有着重要意义。库克表示，对苹果而言，中国是一个巨大的市场，中国也是未来发展战略计划中的关键因素。

jQuery旋钮工具Knob

如果你正在寻找一款显示旋钮的jQuery插件，jQuery Knob是一个不错的选择。它使用Canvas帮助开发者生成旋钮特效，可以使用插件选项或HTML5的data来自定义设置插件属性，方便简捷。

Knobs的主要特性包括：支持只读模式，两个供选择的Callback方法——Change和Release，支持自定义选项并且支持使用HTML5的data属性来配置插件选项，内建不同主题，并且对老浏览器有着不错的fallback机制。





RubyMotion

Laurent Sansonetti是MacRuby的创建者，他在MacRuby的基础上，开发了一款名为RubyMotion的商业应用。利用RubyMotion，开发者可以直接使用Ruby语言来编写和测试原生的iOS应用程序。

RubyMotion应用程序通过命令行进行创建和维护，项目基于Rake工具，可以通过编辑Rakefile进行配置。开发者可以调用所有的公用iOS API，也可以调用第三方Objective-C库或Gems，还可以使用自己熟悉的编辑器。测试则可以通过软件提供的iOS模拟器交互进行。

OpenShift开源

2011年5月，Red Hat在波士顿峰会上启动了其PaaS平台项目OpenShift。时隔一年，Red Hat宣布将OpenShift Origin项目开源。Origin既是PaaS平台OpenShift的开源代码库，也是其社区与项目的名称。

OpenShift支持使用Java、Python、PHP和Ruby语言的开发框架，包括 Spring、Seam、Weld、CDI、Rails、Rack、Symfony、Zend Framework、Twisted、Django和Java EE。它还包含SQL和NoSQL数据存储及一个分布式文件系统。

OpenShift采用对商业应用友好的Apache License 2.0协议开源，这意味着第三方开发者可以方便地利用它搭建适用于自身需求的云计算方案。

Google发布大数据分析工具BigQuery

Google于2012年5月1日发布了BigQuery云数据实时分析服务，它是一个运行在云端的SQL服务，可以对海量数据进行实时分析，客户端不需要做任何事情。

Google于2011年11月对BigQuery进行了封闭测试，现在已经开放给所有用户。它由Google托管，使用Google的计算资源，可在数秒内分析数十亿行的数据，并有群组分析功能。Google认为它非常适用于Ad-hoc分析、标准化报告、数据探勘及网络应用程序等领域。

用户可以通过BigQuery的浏览器工具、命令行工具，或者呼叫多种程序语言打造的REST API来使用BigQuery服务，每月有100GB的免费查询额度，超过后每GB的费用为0.035美元。

Xfce 4.10发布

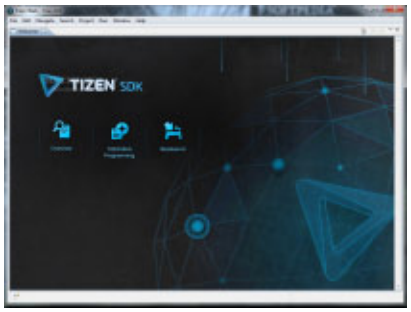
经过近一年半时间的开发，轻量级桌面环境Xfce 4.10发布了，它是取代Xfce 4.8的新稳定版。这个月ESR在The Setup的访谈中还谈到他厌恶臃肿的Gnome而倾向使用Xfce。在4.10版中，开发者的工作主要集中在提升用户体验，该版本的亮点包括以下几处。

全新的应用程序查找器已经完全重写，原有的xfce4-appfinder和xfrun4已合而为一。面板现在有一个可选的竖直显示模式（桌面栏）。不仅如此，Xfce还能多列安排面板插件，这在桌面栏模式中特别有用。全新的MIME类型编辑器让使用者可以轻松地更改用于打开不同文件类型的应用程序。对鼠标、触摸板设置对话框和设置编辑器的功能进行了扩展，可以更好地支持手写板。

现在使用者可以单击鼠标启动应用程序和打开桌面上的文件。此外，4.10桌面可以显示缩略图并通过壁纸列表自动进行。在将窗口拖至屏幕边缘时，可以配置窗口管理器平铺窗口。标签窗口（Alt + Tab）支持更灵活的主题和光标键导航。



Tizen 1.0发布SDK及源代码



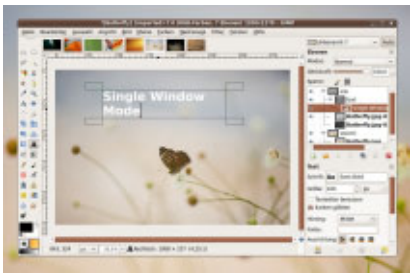
Tizen是一款开源手机操作系统，由Intel和三星主导设计，主要定位于低端市场。Tezen的诞生主要源于Intel在移动市场的失利，以及诺基亚放弃MeeGo转投Windows Phone，而三星的bada并无起色，两者一拍即合，联合设计了Tizen。Tizen被看做是MeeGo的继任者，但与MeeGo不同，其应用基于HTML5而非原生的Linux。1.0版本的发布意味着潜在的设备厂商可以开始实体设备的制造了。

代号Larkspur的Tizen 1.0包含了多项针对SDK和平台的改善，包括增加了基于浏览器的仿真工具，可以模拟 Tizen Web API的工作情况。开发者可以直接在浏览器上运行和调试API，用户还可以在不同平台和设备上模拟Tizen应用，并且可以使用Intel或者OpenGL等硬件加速技术来进行测试。围绕模板和调试除错工具也做出了多项改进，大幅度提升了模拟器性能。

GIMP 2.8发布

GIMP是GNU图像处理程序（GNU Image Manipulation Program）的缩写，是一款免费的图片润饰、制作和处理软件。尽管没有Photoshop精致，但它已内含几乎所有图像处理功能，并且极为灵活。GIMP提供了多种影像处理工具、滤镜，还有许多组件模块，它的接口也相当轻巧。

2.8版本的开发历时3年，将核心图形代码移植到了非破坏性图形处理核心GEGL库中，新增了许多用户强烈需求的功能，主要包括用户界面新增单窗口模式。通过“窗口”菜单的单窗口模式选项，开发者可以选择单窗口模式或默认的多窗口模式，并在退出时保留设置。优化了“保存”和“导出”选项功能，现在只允许将文件以GIMP默认的XCF格式来存储。新增对图层组的支持，并将运行图层以树形结构的方式来排列。将传统的通过单独窗口进行文本编辑的方式改为on-canvas编辑方式。



此外改进了笔刷功能，现在可以通过响应曲线来配置相关参数。优化了Cage变形工具，开发者可以通过可调节的用户自定义的多边形帧，完成对目标对象的弯曲变形等操作。

CSS箭头生成工具CSS Arrow Please

随着CSS3的应用，开发者已可以使用CSS完成越来越多的之前只有图片才可能做到的事情。CSS Arrow Please就是这样一个工具，开发者可以设置设气泡提示位置、尺寸、文字颜色、边框宽窄、边框颜色等属性，CSS Arrow Please可以按要求在线生成不同箭头形状。



Wired

2012.5

A/B测试：改变商业规则的技术内幕

在过去十几年里，A/B测试已然成为高端网页开发的公开秘密，如今它是硅谷在线产品改进的标准方法（但媒体对此却宣传甚少）。通过A/B测试可以对新创意从本质上进行焦点小组实时测试：一小部分用户在不知情的情况下，改道而行，来到与某个既定网页略有差异的页面进行访问，将其操作行为与标准网站上的大量用户的操作行为进行比较。如果结果证明新版本更胜一筹——即能增加点击率、延长访问时间、提高购物量，那就用新版本页面替代原来的页面；如果新版本并无胜出之处，则会悄悄退场，而大多数用户根本不会注意到这个版本的页面。A/B测试似乎使得各种主观设计问题（颜色、布局、图像选择、文本）变成了数据驱动式社会科学的不可或缺的事项。

A/B测试在美国总统奥巴马的竞选活动中显过身手，但这种方法在政治舞台上仍属新事物。不过，它在网站开发上的使用历史却至少可以追溯到千禧年。Google在过去十年中不遗余力地推广A/B方法，它的第一次A/B测试发生在2000年2月27日，那时经常让Google感到困惑的是：搜索引擎在每一页上显示的搜索结果数目应该为多少才最容易被用户接受？就这样Google开始了试验，如今这个数字为10。

A/B测试正在改变各个公司的网站开发方式，同时也在改写一些基本商业原则：一切尽可测试，最终让数字作出决策；以数字的观点替代“最高薪资人观点”；过去的风险在于犯大错误，A/B测试的风险在于使程序员只追求小改进；过去我们通过经验获取教训，如今丰富的数据却让经验教训失去时效。

Google通过倾听数据建立了自己的王国；苹果却说：“知道自己想要什么并不是消费者的事情”；而网络工作者只要尝过A/B测试的甜头，就无法想象离开A/B测试的工作环境。能够测试的网站将更为安全，作选择不再让人举步维艰。为什么B比A更好？不知道。它就是管用。

创造未来的人：《连线》标志人物马克·安德森

他的名字不像盖茨、乔布斯、扎克伯格那样家喻户晓，他的容貌不为芸芸大众所熟知，然而，在他二十年的非凡职业生涯中，要论改变人们沟通方式方面的丰功伟绩，无人能出其右——他就是马克·安德森。他在22岁发明了第一款图形网页浏览器Mosaic，对互联网的推广可谓功高过人，互联网从此走入千家万户；他是Netscape的联合创始人，他让Netscape大规模募股上市，互联网网站在此刺激下急速膨胀；他开创了Loudcloud，以堪称远见卓识的服务将云计算呈现在商业客户面前；最近，他以风险投资人的身份为一系列Web 2.0公司提供支持，Twitter、Skype、Groupon、Instagram及至Airbnb均在此列。

《连线》杂志为了迎接将于2013年1月发行的20周年庆特刊，目前正在进行一系列名为“《连线》标志人物”的专题采访，那些在过去二十年中打造数字文化、向世界传播数字文化的大英雄、急先锋将与《连线》进行深入对谈。选择安德森做头号标志人物再合适不过——他的职业生涯几乎正好跨越了《连线》的历史轨迹，他可以告诉人们如何筹谋未来。采访在位于加利福尼亚的安德森的办公室进行，《连线》主编克里斯·埃德森（Chris Anderson）负责采访，话题包括技术转变以及安德森的五大先知先觉创意。

1992年，创意一：**人人都将拥有网络**——Mosaic和Netscape应运而生；1995年，创意二：**浏览器将成为操作系统**——虽然Netscape败给Microsoft，Google Chrome OS作为一种完全基于浏览器的操作系统却印证了安德森的远见；1999年，创意三：**网络业务将生存在云中**——Loudcloud为此而生，但成功印证这个创意的则是为成千上万商家提供在线支持的Amazon Web Services；2004年，创意四：**一切都将社交化**——为此创立了Ning，同时这方面的成功先行者Facebook、Groupon、Instagram等都得到了安德森的投资或支持；2009年，创意五：软件将吞噬世界——为此创立了Andreessen Horowitz，其远见是“经济因计算的兴起而改变”。安德森相信，

大量科技公司的兴起将改变从零售到房地产、医疗保健的整个经济领域。

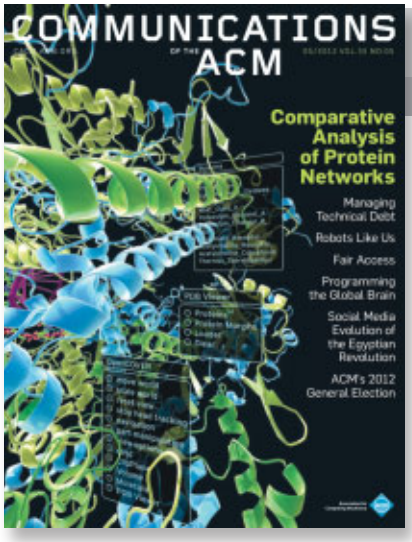
算法写就的新闻稿能否好过人类记者的作品？

先请看这样一幅情景：在芝加哥卢普区，一间宽敞的办公室里，拥有30名员工的Narrative Science公司通过算法设备每30秒左右攒出一篇故事，至于署名则是个哲学问题。这些作品包括激动人心的篮球赛事报道、清晰冷静的公司盈利报告分析、取材于Twitter的喜气洋洋的赛马排名……这些文章并不带机器人腔调，它们或是被发布在《福布斯》等德高望重的网站上，或是用于其他互联网媒体大户（很多大户目前尚未公开）。

Niche新闻服务已聘请Narrative Science帮助其为读者撰写报道。

Narrative Science的写作引擎需要执行好几个步骤。首先必须搜集高质量的数据，这正是财经和体育报道自然而然成为其撰写对象的原因，数据不断波动是这两种报道的共同特点。随后，必须通过算法将这些数据嵌入相应主题的更宽泛理解环境中，Narrative Science因此编写了一系列规则，分别管理每一类主题，公司盈利报告和体育赛事即是其中两类。如何将相应分析转变为文章呢？公司雇用了一组“元作者”，即训练有素的记者，由他们建立起各种模板。这些元作者和工程师一起教会计算机从各种不同角度分析数据——谁赢了比赛？是毫无悬念还是惊天逆转？某位运动员是否表现出色……计算机算法还会综合考虑从其他数据库获得的情境和信息。接下来的一个问题就是架构。大部分新闻都十分公式化，尤其是运动或财经新闻，因此对于元作者来说，构建文章架构相对简单；至于句子，计算机算法用元作者们编汇的词典来撰写。公司将最终完成的产品叫做“记叙文”。

Narrative Science已经熟谙体育和财经新闻的撰写，它认为自己可以做得更多。确实，任何人如果希望对大量数据进行解释、说明，都可以求助于Narrative Science提供服务，慕名而来者也的确络绎不绝，人们愿意花钱把令人困惑的信息转变成好读易懂的文章。不过，新闻记者们目前还不用担心，因为Narrative Science还不打算与他们对着干。



Communications of the ACM

2012.5

技术债管理

1992年，沃德·坎宁安（Ward Cunningham）在OOPSLA会议上发表了一篇报告，提出了技术债的概念，他从不成熟代码出发对技术债进行了定义：“交付初次代码有如陷入债务”。不过，技术债并不局限于由初次代码引发，引发技术债的方法、途径多种多样，可以这样描述：冒着未来耗费金钱或降低过程速度的潜在风险，在当前使用节省金钱或加快过程速度的一切快捷方式。

技术债常常起源于工程设计“最佳实践方法”与其他因素（交付日期、工具成本、工程师现有技术等）之间的紧张关系，简单地说，每当工程师走上缺乏最佳实践方法的捷径，就会产生技术债。例如，在明知生产中需要更好算法的情况下却使用简单而缓慢的算法。程序员在自己的职业生活中或多或少都犯过这类错误——技术债还可能是为了节省时间和金钱而刻意为之的结果。

并非所有债务（无论是技术债还是财务债）都是坏事。在原型中使用简单而缓慢的算法可能是完全正确的做法，前提条件是制订计划，并按计划在代码交付前进行更新。也就是说，要在日程中留出时间、要确保对问题进行追踪，以免在忙乱中错过此事、要知道在执行代码时确实存在能够发挥作用的好算法、要相信管理层会支持你。

如果不对技术债进行管理，则经过日积月累，危机就有可能发生。技术债是不

可避免的，只要管理得当，甚至可能变为好事，但需要小心对待：技术债来源复杂，往往难以预测后果，通常会发生赌未来的情况。技术债管理与风险管理有很多相同之处，可以采用相似的手段来克服。是否理解、沟通和管理技术债可能会给系统的短期和长期成功带来巨大差异。

技术债可以通过多种途径进行观察，也可能在组织的任何层次上引发。只有所有层次都给予支持和理解，才能管理得当，对帮助非技术部门了解债务可能引发的成本提升尤其重要。

为地球大脑编程

综合智能、社交计算……这种种术语代表的是将连在网上的人和计算机合为一体的各种方法，这些方法如今已是头角峥嵘，日后有望成为教育、工业、政府、艺术等一系列领域的真正转机。这些系统将计算机的通信和数据处理能力与人的创造力和高级认知能力结合起来，多年前高深莫测的难题如今解决起来已经驾轻就熟。方兴未艾的技术革新以及目光如炬的广大人群正是这一切的增压器。随着人机网络的规模、范围、连通性不断发展，我们相信，那种认为地球上所有的人和计算机正在构成“地球大脑”的看法将日益彰显其作用。

然而，关于如何为这个地球大脑编程，我们仍然知之甚少。激动人心的成功事例固然有那么一些（如Wikipedia、Google），但大部分应用或是出师不利，或是需要通过大量试验进行精雕细琢。由于人的介入，为地球大脑编程与为传统计算机编程有着显著的差别，动机多样性、认知多样性、错误多样性是其中之最。以认知多样性为例，在大多数计算机系统中，在存储器、速度和设备访问方面，我们面对的多样性范围是有限的。人则相反，他们在自己能够做好的各种工作的各个方面都在不断变化；个体优点充其量也不过是得到片面的了解。这说明，对于在全球大脑这个环境下如何很好地将任务和资源搭配起来这个问题，我们的期望和答案之间会出现定性差异。

为此需要开发强大的新的编程隐喻，正确反映全球大脑中的人机协作方式。例如，如今的创新型综合智能系统包含一整套共用设计模式——采集模式、协作模式以及各种各样的群组和单个决策模式。这些模式反过来又嵌入各种编程隐喻中，如创意生态学、相关网络、智能供应链……一旦这样的全球大脑编程隐喻成为现实，就会反过来从几个方面要求过程：创造“社交运营系统”、定义新的编程语言、传播新软件设计技术。

蛋白质网络比较分析：棘手问题，实践解决

蛋白质是促成细胞生成、决定细胞功能结构的主要因素。蛋白质并非孤立工作，而是相互作用，形成分子结构或传送分子信号。蛋白质图谱及蛋白质之间的相互作用因而成为现代生物学的重大研究难题，在疾病诊断和治疗方面有着重要应用。

自从酵母的蛋白质间相互作用（PPI）数据首次发布以来，许多大规模实验对各物种有机体的PPI进行了测量，蛋白质相互作用数据得以在DIP（Database of Interacting Proteins）等各种数据库中进行积累、评估。尽管如此，PPI数据仍然杂乱无章。各种实验得出的蛋白质相互作用数据的可靠性据估计仅有25%~60%。数据质量不尽如人意促成了跨物种保守性准则的应用，即关注网络中较为可靠的部分，推断出可能有用的组分。由于基本范例是从基因组序列领域借鉴而来，序列保守性（跨物种）往往意味着保守区有可能保持类似的生物功能。这种进化原理催生了一系列以比较多种网络、发掘功能性保守组分为目的的工作，比较层次有两个：蛋白质层和网络层。在蛋白质层上，若蛋白质的网络背景在多个物种中均得以保守，则有可能共享相似的功能；在网络层上，保守子网有可能对应于真正的功能组分，如蛋白质复合体，并有可能具有相似的功能。在这两种情况下，关于任何一个物种的生物知识都可以转移到其他物种，从而有效、正确地解读各种网络。

Nir Atias以及Roded Sharan对比较网络分析领域进行了调查，重点调查了各种计算问题以及解决这些计算问题的种种方法，如启发式算法、参数算法以及基于最优整数线形规划的各种解决方法。他们论证了这些方法在预测蛋白质功能和相互作用、推断蛋白质间相互作用网络在相应功能模块中的组织方式，以及将物种中、物种间的生物过程关联起来等方面的应用。结论是：比较分析工具应用于现有网络数据，能够对细胞功能和细胞的分子组分之间的相互作用得到有价值的发现。

（感谢译者李芳支持）

移动产品速写

Windows Phone Marketplace的软件数量虽然在快速增加，但其中也充斥大量垃圾和盗版应用。清理Marketplace是保持产业链持续发展的必要措施。

大概是因为3月发布了太多新产品，而WWDC和Windows Phone开发者大会还要等到6月，所以这个月尽管重量级的产品并不多。虽说如此，但新品还是不少，我将在本文中对5月移动领域的新品做一个速写。

在超极本领域，反应慢半拍的两大巨头HP和联想ThinkPad终于有了动作，发布了真正意义上的超极本Envy Spectre XT和ThinaPad X1 Carbon。虽然这两款产品未必会带来什么惊喜，但PC和移动设备之间的边界已经越来越模糊，这一切似乎都是Windows 8到来之前的热身。有趣的是，HP的设计副总裁Stacy Wolff在被问到Envy Spectre XT是否太像MacBook Air时说，HP十年前的TC1000也是和iPad一样的平板。这里忍不住吐槽一下，TC1000是一款Tablet PC产品，微软最初设计时，Tablet PC是没有键盘的，到了笔记本厂商那里被生生地加上了键盘，原因是笔记本厂商认为没有人会买一个没有键盘的笔记本。微软吃了个不生产硬件的哑巴亏，直到iPad出来后，各大笔记本厂商才如梦初醒，可惜时机已过，现在又拿出来说事，实在有点搞笑。

最近有点儿热闹的还有一个领域——智能手表，首先是SONY在4月发布了SmartWatch，支持通过NFC和蓝牙连接手机，能够通过手机查看短信、控制音乐播放和拨号功能，还能够作为阅读器和游戏机使用。目前SONY SmartWatch已经开放了SDK，加入了开源音乐播放器和游戏扩展包，开发者已经可以利用SDK来开发手表上的应用了。除了SONY之外，Allerta推出了智能手表Pebble，采用电子纸技术的黑白背光屏幕来显示信息，使用蓝牙与iOS、Android等设备连接，可以完成通话、收发邮件、短信等功能，还可以通过iOS和Android安装应用程序。在这一波智能手表热潮之前，是微软的MSN Direct Smart Watch，该产品也位列史上最糟糕的电子产品排行榜中。MSN Direct Smart Watch采用FM数据广播来进行信息传递，而且能独立完成客户端运算，却没考虑到CPU和电池根本无法支撑如此复杂的运算，惨遭失败也在情理之中。这次的智能手表作为手机配件出现，数据通信、复杂运行都由手机来完成，这样的设计节省了很多的资源，从而减轻了CPU和电池的压力。

微软也在5月针对Windows Phone开发者提出了四项新的规范，严格约束了商标与版权、图标、关键字等方面的内容，原因是Windows Phone Marketplace的软件数量虽然在快速增加，但其中也充斥着海天之类的垃圾和盗版应用。清理Marketplace是保持产业链持续发展的必要措施。诺基亚也将推出自己的杀手级应用，Nokia Reading即将登陆欧洲，为使用者提供本国语言阅读环境，据说还有可能加入相应的支付体系，电子书的店中店呼之欲出。Nokia Drive的下一个版本也在开发中了，新版本可以记录用户的驾车习惯，并且能在Live Tile中显示实时的行车与交通状况，还可以手工调整路线。Nokia Transport会支持87个国家的交通时间表，还支持定位功能，来搜索附近的交通信息。

总之，让我们期待未来更多更新的移动产品吧。



马宁

北京支点联游CTO，OpenXLive开发者。曾任微软亚洲工程院软件工程师，从事Windows Embedded Compact开发工作。加入微软之前，连续四年获得“微软最有价值专家”称号。

个性化的开源操作系统 Haiku OS

相比那些主流的操作系统，Haiku就像一把小巧的瑞士军刀，灵巧迅速地成为工作的帮手。

《IEEE Spectrum》是美国电气和电子工程师协会（IEEE）的旗舰期刊，其文章以报道信息技术领域的最新进展为主。近期《IEEE Spectrum》刊文介绍了Haiku OS的开发和应用情况，引起了不少喜欢尝试新鲜事物的Geeks的关注。

Haiku OS是适用于个人电脑的新型操作系统，这款开源操作系统最大的特点是迅速、高效且易用。Haiku OS的前身BeOS诞生于1996年11月，由Be公司设计并开发。BeOS最初版本只适用于PowerPC平台，两年后发布了能够安装在IBM兼容机的版本。较之Windows在日常办公中的广泛使用和Unix/Linux在Web环境下的普及，基于全图形化设计的BeOS以其出色的多媒体性能在用户群里获得了不错的口碑。值得一提的是，BeOS首次在个人电脑上采用了64位文件系统，为多媒体制作时所需的大规模数据交换提高了运行效率。因此，BeOS被广泛用于多媒体制作、编辑、播放等领域，并被很多专业多媒体开发、制作和设计人员使用。

一直以来，BeOS都在为晋级主流操作系统而努力。然而，BeOS为非专业用户提供的应用程序较少且在使用上存在门槛，致使BeOS的普及发展受到了局限。最终Be公司宣布不再为BeOS后续发展提供支持，并将包括BeOS在内的公司资产出售给Palm公司。就在BeOS被售出的2001年，原先的忠实用户发起了名为OpenBeOS的开源项目，以自由软件社区的形式继续推动他们热爱的操作系统继续发展，并逐渐演变成现在的Haiku OS。

在分析了BeOS优势和局限后，新生的Haiku OS为用户提供了清爽的图形界面，在系统启动时间、计算机硬件兼容性方面有了显著的提升。Haiku延续使用了64位的文件系统BFS（Be File System），并在原有BFS的基础上针对Haiku进行优化，支持基于元数据的文件检索，以保证处理多媒体文件时的效率。此外，采用C++开发的Haiku为应用程序开发者提供了面向对象的API系统调用接口，并与BeOS平台的本地应用程序在二进制和源代码上保持兼容，这使得原先针对BeOS设计的应用程序在Haiku上有了用武之地。

在多媒体展现方面，Haiku全部使用抗锯齿字体、全矢量图形和矢量图标。Haiku还增强了MIDI协议，提供OpenGL支持，并开发了专门的游戏工具包和相应API。Haiku对BeOS的外设输入响应机制进行改进，建立了全局的输入服务来为各种输入设备提供响应，并建立了各项输入设备与操作系统通信的渠道。

2009年9月，Haiku发布了第一个Milestone版本Haiku R1/Alpha 1，并紧接着于2010年5月发布了R1/Alpha 2。在第三个版本R1/Alpha 3于2011年6月发布之后，Haiku第四个版本同时也是最后一个Alpha版即将发布并进入Beta阶段。

不同于Windows、OS X或Linux，Haiku在某些特性来讲是独一无二的。Haiku拥有10秒以内的系统启动速度，运行时仅占用170MB内存空间，且从来不会出现Linux那样内核经常与用户界面不协调的情况。相比那些主流的操作系统，Haiku就像一把小巧的瑞士军刀，灵巧迅速地成为工作的帮手。可以说，Haiku代表用户对个人电脑与众不同的看法，如果你不愿意每隔一两年就因为操作系统更新换代而被迫购买更高性能的电脑，如果你受够了操作系统庞大的体积和与各类软件兼容性带来的恼人问题，如果你认为电脑就应该是一件快速、简单、高效、新颖的忠实工具，那么不妨试一试Haiku OS，或许你会发现它正是为你设计的。



高昂
中国标准化研究院助理研究员，从事信息技术标准化研究工作。关注开源社区，也是OSGeo中国和InfoQ中文站成员。

MySQL——特别响、非常近

作为Oracle的一块“实验田”，MySQL始终能够享受到Oracle产品的技术外溢，帮助它在近2年的时间里，逐步成为一个与“三大”商用数据库等量级的“全能选手”。

本月MySQL无疑成为数据库新闻的主角，5.6版本还没正式发布，其特性就已经成为业内讨论的热点。经过新东家Oracle的大力栽培，MySQL已经从开源旗手变身为真正具有众多“Enterprise”特性的泰坦了。MySQL 5.6预览版，作为一个还在不断演进的版本（DMR，Development Milestone Release）有下面两个特性非常吸引眼球：

改进的复制功能；

在关系数据基础上，增加对NoSQL数据库的快速数据查询。

■ 复制方面，MySQL总算增加了全局事务ID（GTID，Global Transaction Identifiers），凭借它复制集群中的不同数据库，可以通过GTID追踪需要复制的内容，即便某个节点宕掉，其他节点依然可以接手复制工作。以前，用户往往需要通过第三方的复制软件解决这个问题，但5.6版本后，用户将可以通过MySQL自己解决。

这个特征的意义还不限于本地复制集群，因为通过GTID用户可以更加容易地建设异地容灾中心，甚至是建设异地双活（Double Active）的数据中心，因为用户提交的每个事务都可以通过配置好的复制关系进行传送，而不用担心不同节点间因为主键冲突、事务到达时间差异等因素导致的复制冲突和复制回波效应，简化了DBA和应用开发人员针对复制要求的相关工作。

■ 在提高查询性能方面，MySQL也从Oracle“继承”了不少新的特性。

首先就是查询语句和查询结果的组织上，类似Oracle的Package，MySQL的新版本也可以一次性将一批查询作为一个单元提交，并可以将多个查询结果作为一个单元返回，而且查询结果面向前端JavaScript应用做了不错的支持，因为查询结果可以以JSON方式返回。前者可以看作是对MySQL存储过程和函数（从MySQL 5.1开始引入）的一个补充，毕竟它降低了网络往复，比较适合界面相对复杂、业务操作集成性较强的企业应用和电子商务应用；而后者这个特性为很多中间件的开发商提供了一个新的契机，同时也为MySQL的互联网应用提供了非常有效的开发支持，因为随着浏览器前端应用的流行，以往类似的工作常常都需要开发者自己完成，但查询结果的多变性又总是导致这些工作非常繁琐，因此数据库直接反馈JSON结果似乎是面向前端的一个不错的改进，当然这个功能的代价是要消耗数据库端大量的CPU时间，实践中也许会被一些关键应用“毙掉”。

另外，在开发接口方面，MySQL 5.6最吸引眼球（但短期内应用前景还不明朗）的功能就是可以直接访问InnoDB数据库引擎，而不需要访问传统的基于SQL的接口。可以看出，随着Big Data概念的流行，DB2、Oracle、MySQL都在试图提供Big Data＋SQL的“双核”引擎，这不仅是技术发展上的需要，同时也是产品的“卖点”。为了避免其他产品对MySQL市场份额的分流效果，Oracle就适时为它“补补钙”，紧紧把握既有用户，在现有产品框架内满足这批用户对Big Data的潜在需求。而且为了进一步适应Big Data数据典型的一次写入、多次读取的特点，MySQL 5.6支持新的插入操作特性，它允许向NoSQL部分写入数据的同时，不阻塞对数据库索引的访问，典型应用情境中预计会较大降低数据冲突的机会，提高数据访问的并行度。

从MySQL 5.5发布至今，MySQL面临的威胁似乎越来越大，但作为Oracle的一块“实验田”，MySQL始终能够享受到Oracle产品的技术外溢，帮助它在近2年的时间里，在技术能力方面快速成长，逐步成为一个与“三大”商用数据库等量级的“全能选手”。以第三方工具的大量出现为特征，一个围绕着MySQL的生态链正在膨胀，预期它将成为Oracle撬动现有数据库市场格局的一根杠杆。



王翔
软件架构师，主要研究方向为XML、.NET、领域设计和PKI应用。工作之余喜爱旅游、写作和烹饪。

苹果的安全之缺

任何封闭模式运行的帝国，在造就极大繁盛的同时，往往也埋藏着深重的安全危机。

OS X无疑是过去一段时间的安全焦点，这一切要从木马Trojan/OSX.Flashback说起。尽管这个病毒在2011年末就已经出现，但今年发现的新变种利用Java漏洞攻击，在4月上旬达到了感染数量的峰值——部分反病毒厂商监控到超过50万台主机被感染。考虑到OS X只是第二大桌面系统，这已经是一个非常惊人的数字。

同一时期，该平台的后门Trojan/OSX.SabPub也被发现与被称为LuckyCat的APT攻击之间存在联系。这款后门工具利用Office文档进行传播。它位于加州的控制服务器的IP地址（199.192.152.*）曾经在2011年被Windows恶意软件所采用，这可能意味着一些针对性攻击正在转移突破方向。

值得一提的是，两个恶意代码的传播和攻击都使用了Java的一个已知漏洞（编号为CVE-2012-0507）。Oracle在2012年2月底就已经将这个漏洞补上，但苹果公司坚持为OS X维护自己的Java版本，因此该漏洞一直存在。

在类似的安全问题上，苹果公司并不是第一次犯错误。2009年，它就曾因迟迟不修复CVE-2008-5353漏洞遭到了业界的批评。

在如何应对自身安全这一问题上，企业总存在与其根基和文化一脉相承的基因。曾有人评价苹果公司像一个巨大的神秘组织，而神秘组织总是相信自己可以独立地解决自身的安全问题。App Store坚定地维护着不允许发布反病毒软件的原则，也似乎通过人工审核模式造就了无毒的神话。但实际上iBot早在四年前就已经出现。对此苹果公司也许还可以把责任推到“越狱”身上，然而这些年来，App Store上的恶意提交事件尽管远远少于Android Market，但也并非是零纪录，号称“远程越狱”却植入木马的攻击模式更是已经多次出现。

Eugene Kaspersky评价说苹果在安全方面落后微软十年。尽管在部分果粉眼里，可以把这种说辞看成是对苹果封闭政策的不满，但从一个安全从业者的角度，从MAPP的漏洞信息分享群防的机制，到微软CTC等机构对安全厂商的开发支持，我们都能感受到微软为了改善安全性所做的努力。苹果至少没有让我们感受到同样的改善效率和成本投入。

作为一个选择全封闭、不兼容产业模式的商业帝国，苹果公司取得巨大成就在于其对体验极致的追求和近乎完美的产业定位设计。但如果把第三方安全支撑能力完全摒弃在外，一个自身已经成为复杂巨系统的体系，怎么可能具备独善其身的能力呢？实际上，苹果帝国的这种封闭，注定导致其不仅仅在安全上存在问题。比如，因为其封闭，Oracle此前一直没有发布针对OS X的Java7开发工具。

从iPod到iPad的崛起，使苹果不仅摘取了个人智能终端的王冠，也推动了其桌面平台的反攻。同一时间，安全威胁也从病毒、蠕虫时代逐步过渡到了窃密时代。在窃密时代，攻击无论是高度定向的，还是撒网捞鱼式的，都将使任何弱点成为突破口。诸多小众系统和硬件都不能置身事外，何况装机量已经居于第二的OS X和已经在品牌分布上占据王座的iOS平台智能终端产品。面对APT的空前威胁，OS X会成为Rootkit开发和植入的优选目标，iOS智能终端也会成为企业网IT治理的重担和盲区。而苹果自身快速安全响应机制不够健全、对安全厂商高度排斥，如果不求变革，这些都将导致其陷入漫长的一个人的战斗。当年，站在Unix的稳定性和历史底蕴的角度，大量开发者把奚落丢给了微软和Win 9x，而今天，站在微软阵营已经形成的安全架构的角度，这种奚落更多会被丢在苹果头上。

被咬了一口的苹果徽标，在其被设计出来时是作为创意，隐含着对完美的追求；但站在一个安全工作者眼里，或者也可以看作苹果安全之缺的畿语——任何封闭模式运行的帝国，在造就极大繁盛的同时，往往也埋藏着深重的安全危机。如果不拒绝那些善意的力量，那么这种安全危机的不断发酵，也许将使其封闭带来的部分甚至全部收益逐步遭到侵蚀和瓦解。

（编辑注：肖新光指出《程序员》5月刊《探索Duqu木马身世之谜》一文中以下文字有误，“Dvldr（口令蠕虫）和Lovgate（爱门蠕虫）的制造者，就是通过同源性分析判断为同一人”。研究表明Lovegate在版本改进中使用过Dvldr部分已被公开的密码档，但其他证据表明两者没有亲缘关系。经与原作者联系，此处确系笔误。特此更正。）



肖新光
网名江海客，安天实验室首席技术架构师，研究方向为反病毒和计算机犯罪取证等。微博：weibo.com/seak。

OpenStack，火热正当时

OpenStack Design Summit & Conference 2012速递

文 / 程辉



2012年4月5日，OpenStack Essex版本正式发布。按照OpenStack社区的传统，正式版本发布之后就会举行OpenStack Design Summit和OpenStack Conference两场活动，简称“两会”，今年也不例外——“两会”在美国旧金山如期而至。OpenStack Design Summit为期三天，是一场技术Geek盛会，主要面向社区贡献者；OpenStack Conference为期两天，是一场综合性会议，主要面向企业和用户，既有用户案例，又有技术分享。由于新浪云计算在生产环境中部署OpenStack较早，且积极参与社区贡献，所以我也受邀在“两会”上做了技术分享和主题演讲。本文将主要谈谈OpenStack及其社区的现状。

OpenStack生态系统正在形成

2010年7月，OpenStack发布第一个版本Austin。当时只有两个核心项目：计算资源虚拟化项目Nova和对象存储系统Swift。不到两年时间，第五个版本Essex已发布，核心项目增加到6个。新增的两个值得关注的项目是统一认证与授权项目Keystone和网络虚拟化项目Quantum，还有一批正在孵化期的项目和大量第三方开发的项目或插件，如数据库即服务项目RedDwarf、队列服务Burrow等。

IBM、HP、DELL、Intel等IT巨头纷纷表态加入OpenStack阵营，大批互联网企业、传统软件服务商等也开始在测试或生产环境中部署OpenStack。开源项目能获得如此多的支持实属少见。

同时，基于OpenStack的商业模式正在形成。在北美，涌现出一批基于OpenStack的创业公司，影响力较大的是Piston，它立志成为OpenStack领域的RedHat，是几位OpenStack创始人，包括前NASA首席架构师Joshua McKenty于2011年成立的。它提供了Piston Enterprise OS及企业服务，特点是快速部署和安装，只需一个U盘便可在几分钟内将一批裸机架部署完成。其他值得关注的创业公司，如Mirantis，提供OpenStack培训和人力资源服务；SwiftStack，提供基于Swift的私有云存储服务解决方案；Nicira，利用OpenFlow协议和Open vSwitch为OpenStack提供商业的网络虚拟化解决方案，它们的工程师主导了Quantum项目的开发，且将Nicira NVP插件集成到Quantum项目中。

另外值得一提的是HP对OpenStack的商业化尝试。HP Cloud算是最大的一个完全基于OpenStack的公有云项目，经历了半年多的免费测试期之后，终于在五月中旬推出Beta版。

Quantum：OpenStack下一代网络解决方案

在本次Design Summit上，关于网络的讨论是一个热点。在此之前，Nova项目PTL（Project Technical Leader）Vishy已经宣布，目前Nova中的网络部分

不接受大改，只允许bugfix，网络相关的功能将全部由另一个核心项目Quantum实现。Quantum定义网络操作相关的API，并且就像Nova通过插件式的架构支持不同的Hypervisor驱动一样，Quantum也支持多个不同实现，既可以操作虚拟交换机，也支持物理交换机。目前官方支持的驱动有Nova Network模块正用的LinuxBridge和OpenvSwitch，还支持商业的Cisco UCS以及Nicira NVP。Quantum跟Nova Network模块的关系可以理解为Nova Network=Quantum+LinuxBridge Plugin。目前Quantum项目完成了二层网络相关的功能，如创建和管理虚拟网络、端口等。另一个负责三层网络的项目是Melange，它的主要任务是IP地址管理（IPAM）、DHCP、NAT甚至负载均衡。不过未来，Melange将最终合并到Quantum项目中，以补充其三层网络功能，实现大统一，也避免多个网络相关的项目同时存在而造成用户困惑。

Swift存储

在Design Summit上，Swift是一个热门话题。一位来自Nexenta的七十多岁的老太太Caitlin Bester非常引人关注。她讲解关于Swift的多版本支持和排重技术，技术含量很高。Amazon S3很早就支持单文件多版本，但Swift一直没有类似功能。多版本对于互联网业务和数据安全都很重要。同时，倘若文件支持多版本，那么势必大幅增加存储空间占用，因此，这时文件排重就显得尤其重要。

正因为如此，Caitlin同时提出这两个问题的解决方案：对于多版本支持，Swift的Proxy Server通过为每一个PUT新文件的请求，定义一个分布式版本ID来标识，这个ID的生成不依赖上一个版本，并且与Swift的Share-Nothing架构保持一致，不依赖中心metadata存储来生成；对于排重，基于文件块（Chunk）来做，而不是对象级别，客户端（可以是Proxy Server）在上传每一个块时，计算它的散列，然后向系统查询这个块是否存在，如果存在就不必继续上传后面的存储节点；如果不存在则继续。当然，还有很多其他因素需要考虑，篇幅有限，感兴趣的读者可查阅设计文档。

Swift即将发生的另一个变化是PTL John Dickinson提出来的。Swift代码目前比较臃肿，尤其是在Essex开发周期中，大量的新特性、新功能加入到了代码核心中，代码审查人员需要完全熟悉日益增长的代码，给代码审查带来了些困难。因此John提议将Swift周边的功能，如tempurl、formpost、staticweb、rate limiting、swift3、cname lookup等中间件移出来，成为Swift的附属项目。

计费与监控

越来越多的公司开始将OpenStack商用，但都面临同一个问题——不得不自己开发统计与计费模块和监控功能，因为OpenStack官方缺少类似项目。新浪也面临同样的问题，因此开发了计费和监控两个项目，并在Design Summit的一个分会场上分享了计费项目Dough的开发和设计，同时，新浪也在GitHub上开放了Dough的源码，并邀请现场开发者一起参与并改进这个项目。

另外，本次OpenStack Conference期间，亚太地区的OpenStack社区代表举行了两场座谈，讨论亚太地区OpenStack大会的相关事宜。有个重要的议题是第一次会议举办的时间和地点。对于举办时间，与会代表很快达成一致，即在两次OpenStack发行版之间，但对于举办地点，各国代表都希望在本国举办，后来才达成一致——在中国举办。

小结

本次OpenStack两会中，充分感受到了OpenStack一直所宣称的四个Open：Open Source、Open Design、Open Development、Open Community。OpenStack使用的是对公司最友好的Apache 2开源协议。任何人都能参与设计和开发，甚至提出创立自己的项目。只要经过社区讨论，并达成一致，就可以按照你的标准。期待更多的中国公司和开发者参与到OpenStack社区中。



程辉

新浪云计算技术经理，2009年加入新浪SAE，曾担任SAE系统开发工程师和运维负责人。2011年初开始研究OpenStack，并组建团队，成立基于OpenStack的项目SWS（Sina Web Services），为新浪内部和合作伙伴提供IaaS层服务。

旁观 ACM ICPC



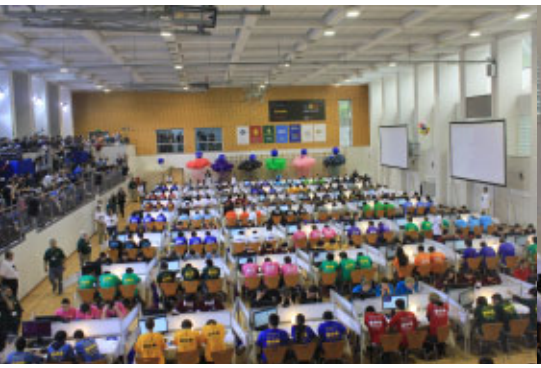
文 / 刘江

闻名于世的ACM ICPC（国际大学生编程竞赛）今年在华沙大学举行，我有幸受邀围观。到波兰的时候，天下着雨，而且连续几天不晴。但坏天气无法掩盖华沙的美景，当然也不会影响到大赛热烈的气氛。在华沙机场，不时可见第36届ICPC志愿者奔走的身影。而坐落在华沙市中心的地标——文化科学宫的外墙，也打上了巨幅的ICPC广告。5月15日举行的欢迎晚宴也在这里举行，众多政府高官出席，规格极高（波兰总统原定出席但临时改变了计划），并有精彩的3D表演。波兰当地的电视、报纸等许多大众媒体也纷纷把焦点集中到了这场世界级的头脑大战上。

看日程才知道，ICPC不仅仅是一场竞赛，它的整个日程长达四天（5月14日-18日），除了正式比赛之外，还有主办方和赞助者组织的各种丰富的学习和交流活动。本次总决赛的主题，定为大数据分析、智慧商务、云计算和社会化商务，应该说很好地反映了目前业界的主要技术热点。作为大会唯一赞助商的IBM，为几百名明日之星准备了体现这些热点的技术演示和技术讲座，使他们有机会较早接触业界最新趋势，更好地规划自己的学术和职业生涯。这应该是ICPC不同于其他纯理论学科竞赛的特色和优点之一。IBM此次提出的观点值得重视，他们认为，人类社会已经进入了一个全新的“智能系统时代”，由计算系统纳米技术、计算、大数据/快速数据处理和学习系统四层组成，不仅能够计算，而且还能感知、学习、预测甚至思考。未来二十年的计算系统将从物理世界收集包含其运行机制的海量信息，从中提取出意义，为人类的各种决策提供可信的参考建议。计算智能将无所不在，计算机将预测我们的期望，而不是像现在这样等待人类的指令，它们将使我们更聪明。

同样热烈的还有编程竞赛社区，TopCoder上已经早早列出了各参赛队及其成员ID，而俄罗斯的Snarknews网站更开始了预测获胜者的场外比赛。赛前夺冠呼声最高的，是东道主华沙大学队，他们的三位成员Tomasz Kulczyński（tomekkulczynski）、Jakub Pachocki（meret）和Wojciech Śmietanka（kefir）都在TopCoder上排名前50，的确令人生畏。而Snarknews则根据历史成绩等数据，将三届总决赛冠军圣彼得堡国立IT与光机大学（简称ITMO）队列为一号种子选手，其中Eugeny Kapun（eatmore）是2009年冠军队成员。近年来中国各代表队成绩极为出色，今年更有近20支队伍参赛。虽然总体上队伍比较年轻，但会不会出现去年浙江大学那样的黑马呢？来自国内的媒体更期待同样获得过三届总冠军的上海交通大学队能够领先一步，获取第四座桂冠。

5月16日晚，我首次在华沙理工大学的晚宴上见到了参赛的所有112支队伍。我很快发现两个有趣的现象：一是各国代表队华裔多，斯坦福大学、明尼苏达大学等三个都是，哈佛大学队也有Neal Wu这样著名的华人选手；二是女生少，去年夺冠的浙江大学队中历史上首次出现了莫璐怡这样的女选手，但并没有改变大趋势，今年总共只有四名女生，来自马来西亚的Fatma Betul Varol因为有头巾非常抢眼，另外在现场还看到明尼苏达大学的华裔Lu Ye，其他两名东欧女选手找都找不到。我还意外地感受到，所有选手虽然都是从全球85个国家、2219所大学的25000名参赛者中经过激烈竞争脱颖而出的，即将投入更为残酷的决赛，但各队之间的气氛非常热络，更像是一个超级大家庭的热情聚会，工作人员、教练、队员大多相互熟悉或者早已久闻大名，大厅里一片老友重逢的寒暄之声。这才想起一位朋友的话：“ACM竞赛







其实是一些老朋友一起进行的一场智力游戏。”我问一名中国选手看好哪支队，他回答说，明年上海交大肯定得冠军……

5月17日上午10点整，总决赛正式在波兰华沙大学管理学院打响，鏖战持续5小时。身着各色队服、各种肤色的选手，十几种不同颜色成堆的气球，一百多支队伍的壮观景象，使整个赛场有一种梦幻般的世界村的感觉。我粗粗翻看了一下发到手里的12道题目，似乎问题B（求多项式积分）和D（求子字符串数）是思路最明显的。

果然，才过了15分钟，斯坦福大学队和去年一样获得了最快解题奖，第一个攻克了B。他们的桌边升起了第一组气球。几分钟后，莫斯科国立和圣彼得堡ITMO也搞定了B，而清华大学和华沙大学先后搞定了D。除了斯坦福外，其他四个恰好就是Snarknews预测排名中的前四，强队还是有传统的。

10:27，台湾大学解出K。而选择先攻L的队伍都遇到了麻烦，只有香港中文大学的同学用时37分钟通过。上海交大也在这里耗费了超过一小时。11点时，圣彼得堡ITMO成为第一支解出三道题的队伍。但华沙、台湾和莫斯科国立也纷纷追上。来自大陆的代表队还没有解出第二道……

将近正午，风云突变：莫斯科物理与技术学院（简称IPT）连续攻克了D、E、K、L，以5道题冲到最前面。圣彼得堡ITMO很快追平。而做出四道题的有9队之多，中国各强队基本都在其中。此时会场里各队桌子上方已经飘满了代表胜利的气球。

到下午1:30时，圣彼得堡ITMO 8道，莫斯科IPT和华沙7道，基本上将其他队伍甩开了。记分板的排名渐渐僵住。1:55，华沙也搞定了第8道！

封板时，圣彼得堡ITMO 8+1（一道未判定），华沙8+2但罚时较多，莫斯科IPT 7+1，上海交大6+2且时间较少。比赛临近结束，东道主和一直领先的大热门冠军之争有了悬念。要知道，去年浙江大学也是在最后时刻才冲出重围，一举夺冠的。历史会重演吗？此时比赛已经结束，选手们纷纷起立，有些媒体进场采访圣彼得堡ITMO，难道冠军已经决出了？

激动人心的最后揭榜时刻，所有人都紧盯各队记分板中pending的题号，被确认通过的队排名将大大飞升：

铜牌：加拿大滑铁卢大学、俄罗斯莫斯科国立大学、日本东京大学、白俄罗斯国立信息与无线电大学。

银牌：白俄罗斯国立大学、中国中山大学、美国哈佛大学、中国香港中文大学。

金牌：中国上海交通大学7道，俄罗斯莫斯科物理与技术学院8道，波兰华沙大学9道……冠军是俄罗斯的圣彼得堡国立IT与光机大学队，他们以解出9道题用时少而险胜！

与此同时，其他成绩较好的中国代表队还包括：北京大学（15），台湾大学（19），清华大学（25），武汉大学（31），福州大学（35），香港理工大学（36），复旦大学（38）。卫冕冠军浙江大学仅名列64。

明年的总决赛将移师圣彼得堡，由本次冠军得主俄罗斯圣彼得堡国立IT与光机大学主办。盼望中国代表队能够取得更好成绩。

（更多图片、决赛试题、解题思路链接请访问本文在杂志官网上的版本。）

选择标准的、代表行业趋势的平台

微软平台战略部总经理Tim O'Brien专访

记者 / 董世晓

现在的IT业，让我们越来越真切地体会到，优秀的开发者俨然成为一种“稀缺资源”，企业对开发者的重视和争夺已经进入白热化的程度，甚至用“得开发者得天下”来形容也不为过。

于是乎，炎炎五月，执掌微软公司开发工具及平台事业部平台战略部门的Tim O'Brien风尘仆仆地来到中国“传经布道”，带来了微软在平台方面的最新观点和进展。

三股力量

在Tim O'Brien看来，IT业界目前有三股力量，对各公司战略目标的制定发生了深远的影响。他相信，这三股力量将会决定到底是谁能够在平台战场上笑到最后。

第一股力量是互联的设备。以前人们多使用单一设备，而现在一个人可以使用多种设备。对于开发者来说，要想抓住在多种设备上出现的机会，就必须把自己开发的应用发布到更多的设备和平台上。显然，这同时意味着工作量的加大。因此，能否帮助开发者使用同一种平台将开发出的应用同样顺利地运行在多个平台上，显得极为迫切。

第二股力量是无间断的服务。当前的云计算时代是与服务密切相关的，而服务又是与后台应用唇齿相依。对于开发者来说，为了提供非凡的用户体验，他们的应用就不仅是要做好前台客户端应用，更要实现服务的无间断性，使前台应用和后台应用畅通衔接。

第三股力量是互操作和开放性。由于之前更多的是在使用单一系统，而现在的多种设备使得异构环境丛生，技术差异性巨大。这便要求要更多地支持开放性和标准化。

三个动作

正是基于上面的三股力量的交相辉映，使得微软在平台战略上的三个动作表露无疑。

第一个是Windows 8。据IDC统计，2011年全球PC、平板电脑出货量分别达3.6亿台、0.69亿台，同时它预测2012年分别会突破4亿台和达到1.06亿台。对于开发人者来说，这意味着巨大的商机：他们可以充分施展和实现自己的创意，在各种各样的平台上开发更好的付费应用或者广告类应用。而Windows 8能够支持开发者使用现有的编程技能，比如C++、HTML5、JavaScript，开发出Windows 8上Metro风格的应用。

第二个是Windows Phone。截至目前，Windows Phone已经拥有八万多个应用和十万多名开发者，并且和HTC、诺基亚、三星、中兴等手机厂商建立了合作。和Windows 8一样，编程人员掌握的不管是C#还是其他语言，都可以轻松使用Visual Studio来开发Metro UI风格的应用。

第三个是云计算。在Tim O'Brien此次走访中国企业的过程中，他发现企业的技术负责人已经意识到，计算、存储、网络等IT基础设施已经是商品化的，不应该浪费过多的人力和物力在这上面，而是去做一些更富有创造性的工作。与其他厂商不同，微软云计算战略强调的是全面覆盖的、端到端的整体的云技术和工具：在终端部分有Windows，后台部分有.NET；私有云方面有Windows Server，公有云方面有Windows Azure；同时，应用开发基于.NET，且使用同样的工具——Visual Studio。这一点对于国内开发者来说非常重要，因为云计算不是简单的买过来就算完的东西，它代表着一种态度、一种方法——一种全

新的系统结构和应用开发的方法。不管何种类型的客户，政府类型或者商业类型，都可以利用这种方法部署私有云或者公有云服务。

两个疑问

第一个是平台选择。经过这些年的积累，市场上有了很多平台，有开源的、不开源的，又有针对各种不同设备的，而云计算时代又带来了更多的PaaS平台——既有传统的Google App Engine、Microsoft Windows Azure，又有Amazon Elastic Beanstalk、Salesforce force.com这样的原本只注重IaaS和SaaS的厂商在PaaS平台上的尝试，更有开源PaaS平台VMware Cloud Foundry为代表的后起之秀。平台多了，确实给开发者提供了更多的选择，但同时也让他们陷入了到底选哪个平台的烦恼。人都是这个样子的，不是吗?

对此问题，Tim O'Brien的建议是，虽说平台的选择问题是不可避免的，但开发者应该将自己的技能向比较标准的、代表行业趋势的平台上靠拢，可以使用类似Visual Studio 11的跨编程语言开发工具，以能够最快地开发出真正针对不同设备的可移植的应用。

第二个是开发能力扩展。经过这些年的发展，Windows积累了相当庞大的开发者群体和应用。当前，开发者面临的一个突出的问题就是如何将Windows平台上的开发能力，拓展到Windows 8和Windows Phone平台。

Tim O'Brien认为，如果认为一个应用只是做了很小的改进，就移植到其他设备上的话，是一种错误的认识。因为不同类型的设备有其自身的特点，很多应用都是针对设备的特性而开发的，因为只有这样才能最大发挥出设备的体验。

对此问题，微软大中华区副总裁、首席云计算战略官谢恩伟补充道，微软将着重于如何帮助开发者更快地把应用带进市场、如何对于已有的Windows平台支持做更好的延续性使用，以及如何帮助开发者选择最好的开发工具来提高效率。在Windows Phone平台上，微软已提供了一系列第三方平台的SDK，使开发者无需开发一些常用



Tim O'Brien访华布道微软平台新战略

功能，直接使用SDK就可以进行二次开发。在技术延续性上，一方面，Windows编译方面的能力是以往延续应用下来的，并不是全新的技术，因此应用的拓展相对容易；另一方面，开发者如果掌握了HTML5和JavaScript等跨平台技术，就可以在不同平台上进行沿用。

一个结论

从某种意义上来说，2012年是对微软至关重要的一年——Visual Studio 11已经推出，Windows 8呼之欲出。毫无疑问，在移动设备操作系统方面，微软已然处于追随者的境况。虽说Windows 8被认为是继Windows 95以来的微软操作系统的最大变革，肩负着为微软在移动设备操作系统战场上摧城拔寨的重任，也确实得到了不少好评，但如何在IT消费化和移动化的发展大潮中，更好地吸引开发者使用新工具（Visual Studio 11）开发新平台（Windows 8）上的应用，并进而扭转起步晚的劣势以取得后发制人的成果，仍然是一项艰巨的工作，离不开微软整个平台战略的支持。中国作为全球开发者聚集的重镇之一，自然成为兵家必争之地。Tim O'Brien在这个时间访华，也恰好印证了这一点。他的“选择标准的、代表行业趋势的平台”观点，对于开发者来说，无疑是受用的。

产品新人的算术课

需求规划能力修炼

文 / 顾晶晶

分析用户需求、为产品制定特性规划和发展方向是产品经理的主要职责，也是最考验产品经理内功的一个环节。在腾讯、百度等公司里，一个产品新人在进行了背景知识的调研和培训之后，从第一次版本规划才算开始了真正的工作。产品经理的技能在学校里是没有教的，虽说人人都是产品经理，但人人都是从零开始，因此起步这一年左右的时间不但是产品新人的阵痛期，也是成长最快的黄金时代。我将在本文分享如何在这个阶段做加减法，供0~3岁的产品经理共同讨论。

第一次版本规划，做好减法

一般来说，产品新人起步会从负责一个大产品的小型模块开始，或者在像腾讯研究院这样的环境中从负责一个小型产品开始。这个产品往往来自已有的项目，并非由新人从头调研、申请立项。而充满热情的产品新人大多积累了各种新奇、有创意的产品思路，迫不及待地想大干一场。在版本规划时，觉得什么都可以做：时下红火的功能为什么不做呢？现在流行的设计为什么不采用呢？洋洋洒洒写完文档之后，结果却在需求评审会上焦头烂额，被PK秒杀。因此在制定第一个版本计划时，要先尽可能地熟悉产品，谨慎地做好减法。

- 产品的核心价值是什么？减去不核心的创意功能。
- 主要的目标用户是谁？减去不属于目标用户的场景猜想。
- 用户的核心诉求是什么？减去不一定存在需求的功能设计。

这其中的追问和体会是最重要的，能帮助你真正明白是“谁在用这个产品，他们的核心诉求是什么”、“没有你的产品时，他们这个诉求被满足的程度如何”。

然后再来判断你想要做的功能，是诉求的解决方案吗？如果不是，那这个你为之激动的功能在现在的用户群中，可能并没有水花；如果是，那它为什么能更好地解决他们的问题？举一个例子，QQ影音有一项功能，可以通过WiFi从PC向iPad的QQ影音无线传输文件，目的是为了帮助QQ影音iPad版的用户解决只能通过iTunes连线同步进行文件导入的麻烦。最终这个功能做得非常轻，仅在QQ影音iPad版中打开开关，就可通过PC版的QQ影音完成选择和传输文件的过程了（如图1）。

但在设计这个功能时，曾经开发了一个完整本地文件管理功能的实现方案，包含视频文件夹展示、常用文件列表等功能，希望满足“用户在观影和管理文件时产生的传输需求”。但仔细分析QQ影音和QQ影音iPad版用户的习惯可以发现，传输文件的需



图1　利用QQ影音向iPad传输文件

求其实是在用iPad观影前，场景并不是之前猜想的那样，因此不需要一个看起来美好但繁重的本地文件管理功能。

在长远规划中，做出加法

经历过半年到一年的稳定产品迭代之后，对产品的前世今生、现有环境有一定把握之时，产品经理可以去关注需求中更广阔的天地。但在专注一段时间之后，产品经理可能不再像开始时那样，对所有的新鲜内容都有热情，容易陷入自己从事的领域内，眼界变窄。这时需要的反而是提醒自己，要开拓眼界看交界的产品，去尝试新生的产品形态、交互方式、商业模式。此时要思考：

- 产品解决的基本需求是什么，解决得如何？
- 这一基本需求实际归属于人性根本诉求中的哪一类，是否被关注到？
- 这一类根本诉求中，还有其他哪些基本需求，是否有机会扩展？还有其他哪些应用在这一领域？是否可以合作？

微信等一些产品成功之后，人们开始更多地关注人文在设计中的力量。这里并不是要强调情感输出，这对产品新人来说是刻意便会矫情的目标；而是想说产品经理在修炼基本功的同时，把对产品的热爱融入自身的洞察力、价值观的提升之中，才能在快速发展的环境和随之变化的用户使用场景中，对创新点保持敏感。

我相信任何一款产品，包括工具类产品，背后的需求都可以投射到马斯洛需求层次当中。例如QQ旋风和QQ影音，都是工具类产品，满足人们高速下载、流畅观影的需求。而这之外，其实是用户强大的娱乐诉求，人们在使用QQ影音时，心中想要的是放松、消遣。所以虽然是强技术型产品，QQ影音的设计，尤其是iPad版和Android版会采用轻松休闲的场景化设计，把复杂的双核解码、QQ旋风离线下载的结合、WiFi传输都简化成几步必要的操作。同时将找资源、下载、观看三大块需求封闭地结合在一起，关注当用户想看一个私人收藏的电影电视时，他们真正想要的是什么。

大道至简

对产品特性的拿捏，是产品经理永恒的课程，你常会看见一群PDM坐在一起讨论这个话题，乐此不疲，却也总不能得出完美的答案。分析用户需求、把握团队呼吸、领会行业动态，再融会贯通，琢磨用户体验才能得出一二。



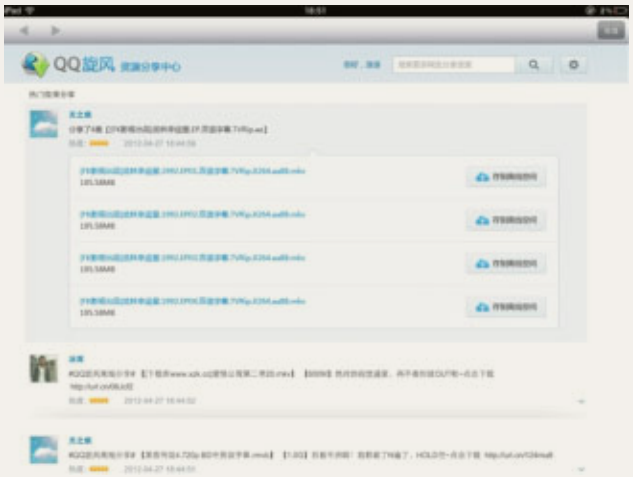


图2　QQ影音还结合了离线下载和双核解码功能

好在我们可以永远追寻更好地满足用户的需求，把“为用户遇到的问题提出解决方案，帮助他们更好地享受生活，哪怕只是一点点”作为品鉴产品设计和规划的准则。不用担心你所做的产品不似那些拿到千万风投的产品一样能说成一个完美的商业故事，市场的繁荣可能有泡沫，但反馈和访谈里的用户需求是实实在在的。

看山是山，看水是水；看山不是山，看水不是水；看山仍然山，看水仍然是水。

窃用此句，与大家共勉。



顾晶晶
腾讯研究院产品经理，专注于PC&移动客户端产品设计和运营。



产品经理

本期封面报道，分享了多位一线产品经理的成长心路和实战经验，涵盖优秀产品经理所应具备的素质和技能、产品经理该如何培养逻辑思维、如何将用户变成布道者、如何更好地解决与技术人员之间的意见分歧、如何进行更有效的市场调研等多个热门话题，更有产品经理常犯的错误和如何警惕恶性循环等来自实践的反思和总结……

优秀产品经理必备的素质和技能

文 / 张月

产品经理是一个看起来颇为风光的职业，我身边有很多开发工程师和设计师都将产品经理作为自己的职业发展方向。总结起来，原因大概有以下这些：产品经理对产品具有一定的掌控力和话语权，可以调动资源为自己的目标服务；如果产品在市场上取得成功，产品经理能够享受到比其他人更多的成就感；产品经理所需的软性技能比硬性技能多，只要有想法并愿意付诸努力，就有可能达成目标。

但同时，产品经理要在项目中充当迷你CEO的角色，就好像是冠军球队的中场指挥官，对人的综合能力要求较高。想成为一名优秀的产品经理，并不像很多人想象得那么简单。优秀的产品经理究竟需要具备何种素质和技能呢？下面我结合自己的经验和知识阐述一下。

打造优秀产品的热情

“不想当将军的士兵不是好士兵”。我常对身边的产品经理说，一旦选择了这个方向，那么自己职业生涯中所取得的成就必然与曾经参与过的产品所取得的成绩息息相关。假如在十年之后，历数自己曾经参与过的产品，没有一个足以留下来并让人们记住，着实是一件令人悲伤的事情。不仅自信心会受到严重的打击，而且将来的职业发展空间也将受到限制。

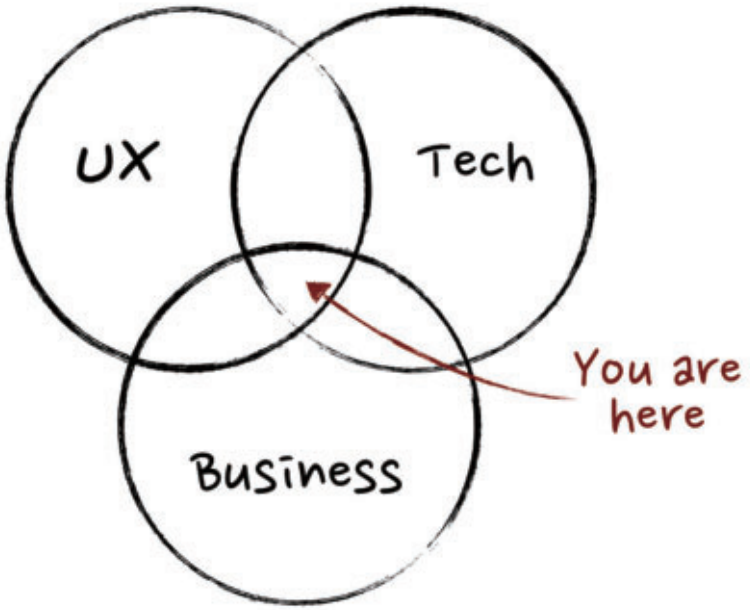
产品经理不能满足于平庸，必须具有打造优秀产品的愿望和激情。现在是一个产品同质化非常严重的时代，创造新产品的成本在不断降低，竞争比以往任何时候都激烈。只有卓越的产品才能够脱颖而出，形成竞争优势。

当脑海中浮现出假想的产品轮廓时，有种兴奋得无法入眠的感觉，这才是设计产品的最佳状态。然而，产品经理不仅要带着激情来设计产品，更要将打造优秀产品的激情传递给团队的其他成员。并不能要求每个人都能够成为乔布斯，但如果总对自己所负责的产品缺少热情，那么很难成为一名优秀的产品经理。

战略思考能力和学习能力

很多知名互联网公司喜欢通过校园招聘的形式来招募工程师，这是因为校园中的学习和锻炼可以为优秀工程师所具备的基础能力打下坚实的根基。而对于产品经理职位，一般公司都要求至少具备3~5年的相关工作经验。这是因为，产品经理需要了解用户，具备市场分析能力，从宏观层面考虑产品的战略方向，而这些往往是在校园中无法学到的，需要在实践中学习和积累。对产品经理岗位感兴趣的大学生应该提早积累自己的专业知识和经验。选修市场营销方面的课程，对开始产品经理的职业生涯绝对有所帮助。

在新浪微博上的一篇热帖中，有人说：“产品经理在一天中有30%的时间在网上到处‘溜达’，浏览网上正在内测、封测、公测和正式发布的各种产品；20%的时间在QQ、MSN、微博上闲谈、求码……”这可能会对处于职业生涯早期的产品经理适用，因为产品经理应该具有强烈的求知欲，在“溜达”和浏览的过程中，培养自己对产品的感觉，逐渐适应快速变化的互联网。



对技术趋势的了解和把握能力

产品经理需要什么样的技术背景，是一个非常有争议的话题。不同的业务模式，对产品经理的技术功底要求也不尽相同。如果在Google这样以技术为核心竞争力的公司中担任搜索业务的产品经理，必须具备扎实的技术功底；而在媒体或社区类型的互联网公司，则通常在技术方面对产品经理候选人没有硬性的要求。

我认为，要想成为优秀的产品经理，一开始可以不懂技术，但不可以从内心对技术具有排斥心理。要努力弥补自己技术知识的不足，尽可能多地掌握网站背后的实现原理，并在此基础上，关注前沿的技术趋势。这样，自己的职业生涯一定会从中受益。这就像一位优秀的建筑设计师，应该了解工程力学和材料学的知识：一方面，有利于与工程师进行顺畅的沟通；另一方面，对于把握需求尺度、作出正确的判断也有很大帮助。

就我而言，尽管整个职业生涯中从来没有担当过开发工程师的角色，但对技术的兴趣驱使我紧密关注前沿技术。在早期的一些创业经历中，我甚至独立承担了从网页制作、图形设计、系统维护到程序开发的全部工作。虽然当时感觉很累，但在后来从事产品设计和管理工作的过程中，都从先前积累的技术经验中受益匪浅。

不过，对于一些工程师出身的产品经理，却要避免无意中混淆自己的角色，过度从技术实现角度来思考问题。这不仅容易造成产品经理与工程师之间的矛盾，还会分散自己的精力，忽略本来应该更加关注的市场需求、用户体验和数据分析等方面。

审美能力和细节把握能力

对于从设计师角色转换到产品经理生涯的人来说，审美能力并不是问题。但对于一些有工程师背景的产品经理来说，这往往是容易被忽略的方面。他们比较重视产品中逻辑的严谨性，但设计出的产品往往存在易用性上的不足，缺少打动用户内心的能力。逻辑的严密固然重要，但也不可小视产品中应该具备的人性化因素。细节决定成败，做好产品需要站在用户角度换位思考。要清空自己站在设计者角度上的立场，重视每一个细节，用心去感受用户的体验。

优秀的产品经理要善于发现生活之美，努力培养自己在艺术方面的兴趣，如摄影、书法、绘画等，提高自己的艺术修养。这些看似与业务无关的事情都会有助于产品经理做好本职工作。

产品运营和数据分析能力

很多产品经理会在开发阶段投入很多精力，而产品一旦上线之后就松了一口气。事实上，这种做法非常不可取。产品始于上线，赢于运营，永远不要忘记产品的成功应该以获得目标用户群的认可为标志。特别是对于已经上线的产品，应该极度重视运营数据的积累和挖掘，并由此掌握用户是如何使用产品的。

说到运营，这是一个很容易被误解的概念。很多人认为运营就是靠人为的手段来拉动网站的流量、带动用户的活跃度，具体体现为制作专题、组织线上活动、微博和论坛话题炒作等。而事实上，真正的运营应该建立在对数据的深度分析和对用户群深入理解的基础上。优秀的产品经理应该具备运营产品的能力，并且能调用各种资源来

帮助自己达成目标。

我比较欣赏豆瓣阿北对运营的理解。从狭义上讲，豆瓣是一个不强调运营的公司。而从广义上讲，豆瓣是一家极度重视数据分析和挖掘的公司，用数据来分析和验证每一个产品决策，根据分析结果来提升用户体验，通过产品和技术来达到运营的目的。进一步讲，重视数据而不被数据役使，用数据启发而不是数据驱动式决策，这才是最高层次的运营。豆瓣曾经先后关闭了豆瓣旅游和豆瓣说两个产品，一方面源于对数据的观察，另一方面也来源于自己的战略思考。

团队合作和需求管理能力

表面上看，产品经理对产品设计和运营具有决定权。而实际上，产品经理这个职位的特殊性在于，本身并不具备很强的行政权力，更多的是要依靠自己的专业权利。这就需要产品经理在与团队成员合作的过程中，不断储蓄自己的“信用”。积累的“信用”越多，在项目中的影响力越大。

除了具备过硬的专业素养之外，产品经理一定不能是刚愎自用、忽视别人感受的人。否则，永远无法积累足够多的“信用”，在项目执行中也会举步维艰。

产品经理应该同时兼具艺术家和管理者的气质，善于在理想和现实之间寻求最佳的平衡点，作出理性的决策。在产品开发过程中，最容易犯的错误之一就是把握不好项目的需求规模，导致开发周期超出预期，为了赶进度而让产品仓促上线。当然，不是说每个产品都需要做到完美的地步才可以上线，但在用户体验越来越重要的现在，必须要坚持自己的产品达到基本的质量标准。

产品经理要懂得且舍得对自己的产品做减法，一开始不要追求大而全，要在无数个功能点中找到最核心的那些，并将它们做到极致。苹果就是信奉这种产品设计哲学的成功典范。例如，第一版iPhone刚刚推出时，相对于竞争对手的产品，在很多功能点上都是有差距的。例如，不支持3G网络、通话质量不佳、相机成像效果一般、电池寿命太短等。但正是由于苹果敢于大胆给产品做减法，才得以在触屏体验上取得了革命性的突破，

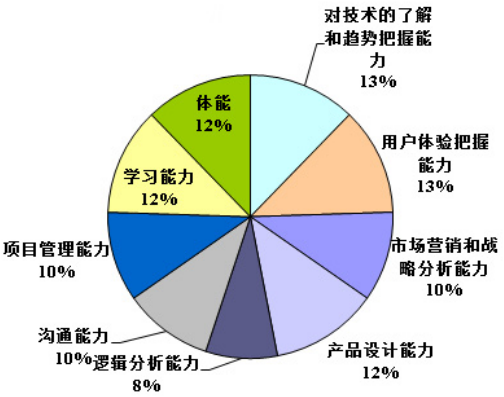


图1 产品经理所需的各种能力

并因此在手机市场上树立了独特的品牌形象。这给苹果赢得改进产品的时间，从此每隔一段时间就会推出一个新版本的iPhone，逐渐让原来产品中的不足之处得到了大幅度的改观。

优秀产品经理必备的能力，还有产品设计能力、逻辑思维能力、项目管理能力等。由于篇幅限制，这里不予以详细讲述。最后还要强调一种专业以外的能力，就是体力。经常锻炼身体，保持乐观、积极的心态，才能承受住高强度的工作压力。毕竟，对于产品经理来说，生活和工作的界限往往不那么明显。

最后，我将产品经理所需的各种能力总结为一张示意图（如图1所示），供各位读者参考。做一名优秀的产品经理，你准备好了吗？



张月

优士网联合创始人兼首席产品官，资深互联网产品设计师，在互联网领域有超过10年的从业经验，对互联网商业模式、用户体验和产品设计都有着深刻的了解和精准的把握。

产品经理的逻辑思维

文 / 吴伟

逻辑思维是对产品经理最基本的要求之一，一个合格的产品经理必须具备良好的逻辑思维能力。良好的逻辑思维能力能够让你在与他人沟通时更加有条理，也有助于设计出更好用的产品。对于入门级的产品经理来说，培养自己的逻辑思维能力尤为重要。

下面是入门级产品经理的一些常见特征。

- 你的日常工作通常只是把领导的想法落实为具体文档，而不是独立地设计产品。
- 你常常觉得无法跟工程师沟通，他们不听你说的，他们说的你也听不懂。
- 你被同事抱怨不专业。
- 你知道自己能力有欠缺但不知道如何提升。
- 你常常觉得自己考虑问题不全面，产品文档丢三落四。

如果你符合其中2项或更多，说明你属于入门级产品经理，而本文将有助于提高你的逻辑思维能力。

在组织语言方面的逻辑思维

下面两段话，哪一段让你感觉更好理解？

第一段：用户首先登录，对于未注册用户也可以不登录直接浏览。用户进入首页，点击“数据分析”进入数据分析界面。用户可在此界面添加股票和财务指标，有6个默认的财务指标，用户也可以自行添加指标。用户在添加完股票和财务指标后，界面可以展示每只股票的相应数据，用户可以通过比较这些数据决定如何投资。

第二段：本网站可以帮助用户通过数据对比决定如何投资股票。用户在“数据分析”界面添加股票和财务指标，即可展示每只股票的相应数据。“数据分析”界面默认展示6个财务指标（用户也可以自行添加指标）。用户分为注册用户和未注册用户两种，对于注册用户登录后进入首页，点击“数据分析”进入数据分析界面；对于未注册用户，也可以不登录直接浏览。

看完第一段再看第二段有没有仿佛先看谜语再看谜底的感觉呢？因为第一段话是按照自然顺序来描述的，而第二段话是按照逻辑顺序来描述的。

现在我们一起来看看怎样把一段自然顺序的话转化为逻辑顺序。为了便于说明，我把第一段话的每个分句都标上序号。

①用户首先登录，②对于未注册用户也可以不登录直接浏览。③用户进入首页，④点击“数据分析”进入数据分析界面。⑤用户可在此界面添加股票和财务指标，⑥有6个默认的财务指标，⑦用户也可以自行添加指标。⑧用户在添加完股票和财务指标后，⑨界面可以展示每只股票的相应数据，⑩用户可以通过比较这些数据决定如何投资。

通读完这10句话之后，我们会明白，这个网站最核心的功能是帮助用户选定一只股票来进行投资，而用户选定股票的方法是对比数据。因此，**帮助用户通过数据对比决定如何投资股票**是这一段话中最核心的意思，应该放在最前面。

我们还知道，用户进行数据对比的方法是第⑤句“用户可在此界面添加股票和财务指标”和第⑨句

“界面可以展示每只股票的相应数据”，所以这两句要放在第二的位置上。

之后我们看到财务指标是有默认的和自行添加的两种，而无论是默认指标还是自行添加指标都不会对用户对比数据造成本质影响，它只是这项功能的辅助功能，因此第⑥⑦两句可以放在第三的位置上。

最后我们发现，用户也分为两种，而两种不同的用户对前面的功能并没有实质影响，只是操作步骤略有不同而已，因此①②③句应该放在最后。

经过这样一番分析和整理，我们再把语句中个别词汇调整一下，让整段话完整通顺，就是第二段文字的内容了。细心的读者应该发现了，第二段话里关于用户的描述多出一句“用户分为注册用户和未注册用户两种”，为什么要加这么一句呢？这是因为默认情况下，前面的句子在逻辑重要性上要高于后面的句子，但注册用户和未注册用户是平等的两个分支，因此需要添加一句说明告诉读者：后面的句子可不再是从重到轻的顺序，而是平等的关系。

当我们与他人沟通或撰写文档时，一定要对自己想说的话思考一番，按照逻辑顺序表达，这样能让听众听明白、观众看明白。

在设计产品方面的逻辑思维

作为产品经理，我们不可能为每个用户当面讲解产品如何使用，或者要求他们先读一遍条理清晰的说明书再开始使用产品。我们必须让产品本身就符合逻辑，让用户一看就懂，无需学习就会使用。

这就需要我们首先为产品设计一个符合逻辑的信息构架，而页面则是对这种逻辑构架的具体体现。

看看下面的例子：请尝试从图1的页面中找到产品名称、价格，并将产品加入购物车。



图1 页面一

有没有感觉完成操作有点困难呢？那么再对图2的页面进行同样的操作。



图2 页面二

这次感觉容易些了吧。为什么两个页面功能类似，布局也差不太多，会给用户造成如此不同的使用体验呢？因为页面一的信息架构设计不符合逻辑，而页面二的信息架构逻辑性很强。

我们先展示一下正确的信息架构，大体上可以用图3表示。

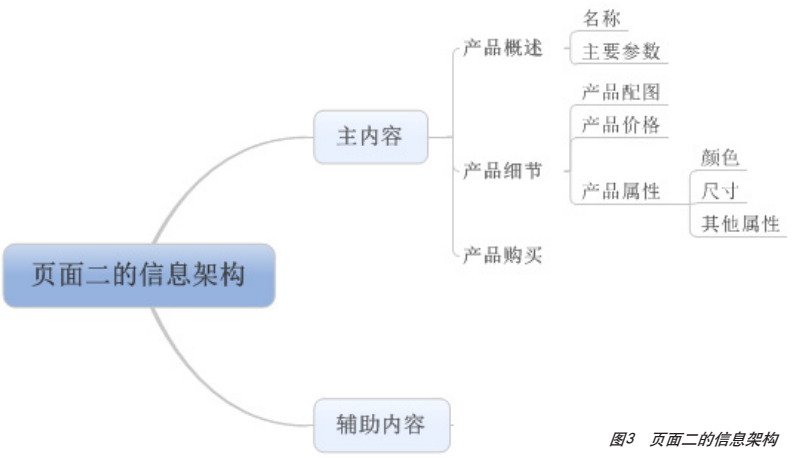


图3 页面二的信息架构

在正确的信息架构的基础上，页面二表现得自然很合理，如下所述。

- 将与所购买的商品直接相关的内容放在页面左侧；将买家描述（辅助信息）放在页面右侧，并且加底色做明显区分。这样，用户无需阅读内容，从布局上即可看出来左右分布的是不同的内容，并将注意力集中在左侧主内容上。
- 左侧主内容呈现了如图4所示的布局结构。

产品细节采用了左右排版，产品概述宽度可以同时覆盖到产品截图和价格、运送地等相关信息。这是因为主内容必须布局成一个封闭的矩形才让用户能够看出来这是一个整体。



图4 页面二左侧主内容的布局结构

即使产品概述的字数很少，也必须保持留白，不要让下方的内容顶上来。这样，主内容部分在视觉上依然可以保持封闭的矩形。否则就会如图5所示，主内容被分割为两块，给用户造成困扰。



图5 主内容被分割成两块

■ 立刻购买和加入购物车两个按钮放在产品细节下方。按照信息构架的逻辑，此按钮也属于主要内容，自然要放到主要内容部分里。

■ 还有一些细节，页面二也做了比较合适的处理。如产品概述里对产品参数的取舍，怎样保留那些用户最为关心的参数而使得页面看上去更简洁；对于价格部分的处理，使用大字号、红色字，而其他文字全部使用黑色，这样就会让价格的数字更加突出，更便于用户查找。

下面我们再来分析一下错误的（如图6所示）逻辑构架。

■ 错误一：将无关的内容放到了一起，而将关联性很强的内容分开。

例如，在中间部分的正下方有一个飞利浦的广告，这与整个产品并没有任何直接关系；左侧的产品配图与产品概述的高度平齐，这使得产品名称看起来似乎并不是在描述配图上的产品。在页面右侧（购物车的下方）又有两个相关商品的推荐。这样就要求用户必须仔细阅读完整页面才能知道哪些信息对自己有用，哪些没用。

■ 错误二：在产品概述里加入了太多的东西。

页面一把CPU主频、内存等这些详细的参数都加到了概述中，使得概述变得冗长。

■ 错误三：在价格附近放了太多的彩色字，使得用户根本无法以最快的速度找到价格。

可见，缺乏逻辑思维会降低产品的使用体验，甚至导致用户流失（页面一是亚马逊中国，页面二是淘宝）。

不断提高自己的逻辑思维能力，不仅可以在工作中如鱼得水，在生活中受益匪浅，而且当你用逻辑的眼光看待世界时，过去某些复杂的事物也开始变得简单。

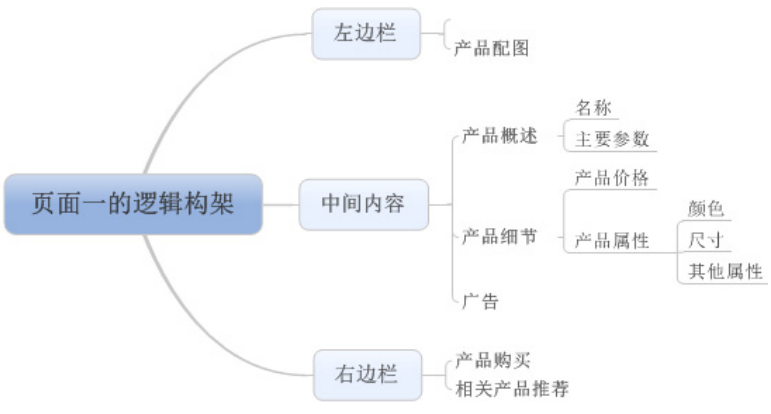


图6 页面一的逻辑架构



吴伟

约瑟网络科技有限公司产品总监。曾在新浪担任产品经理。

如何将用户变成布道者

文 / Rohan Jain 译 / 王江平

如果问一名苹果用户：“为何如此喜爱苹果？”得到的回答通常是：“苹果产品强大稳定、时尚美观，非常好用。”然而与普遍的公众信念大相径庭的是：这并不是人们喜爱苹果的原因。诚然，苹果的设计时尚美观，产品强大稳定，但世界上诸多其他产品同样如此。为什么偏偏苹果的用户极致到无以复加？设计和美感不足以解释为什么苹果用户情愿在寒风中站立一宿，只为排到队伍前端去购买一款新产品。设计和美感也不足以解释为什么苹果用户不想用PC做任何事情，即使出现新款PC能以更低廉的价格提供美妙的特性。

苹果用户之所以成为苹果公司的布道者，并非因为产品的时尚和美感，而是源于所有苹果产品的理念：不同凡想（Think Different）。你见过苹果的广告吗？苹果甚至很少做产品秀。它向人们展示的是在分享苹果的信仰，展示的是苹果在如何挑战现状。它向用户展示是谁在“不同凡想”。**更重要的是，苹果的所有产品都是这一信念的反映。**通过不断创新，苹果公司一直在提供与主流产品相比真正做到“不同凡想”的东西。

苹果的企业领导人奉行与公司一致的信念，这并非巧合。乔布斯是个与主流格格不入的人。他一生所做的事都在重复着“不同凡想”这一信条。事实上，你甚至可以说这一信念也是他死去的原因之一。他从未认同社会主流的行事方式，因而不愿通过常规疗法医治自己的癌症；他相信纯素食谱这种东西有益于治疗自己的病症。乔布斯一生做的每件事都在挑战现状，而大家也都清楚这一点。因此，苹果的每位员工都明白乔布斯的主张，并确保苹果的每款产品都真正挑战现状。

iPhone、iPad，甚至苹果 II，全都一样。它们只是不同的产品，体现的却是**完全一致的理念**。

我想说的是，如果想把客户变成布道者，那么就要确保所做的、所创造的一切都与公司的目标保持一致。如果你是公司的领导人，那么公司的每位员工和每位客户都要真正相信你做的所有决策都符合公司的目标。

如果作为公司领导的你过着一种体现公司意义的生活，同时确保所有公司决策都是这一信条的反映，慢慢地，用户就会加入你的征程。用户将开始帮你宣传，由衷地相信你的公司就是他们人格的代表，开始相信你总能代表他们的最大利益，这一点甚至无须向你询问。这般忠诚无关设计或特性，而只关乎社会性物种（人类）与生俱来的需求——加入代表其价值的团体。这就是为什么人们会在机场极度自豪地开启自己的苹果笔记本电脑，乐意向整个世界展示它。对这些用户而言，苹果已经不只是一家公司，更是一个代表他们是谁的符号。这些用户觉得自己从属于一个挑战现状的团体，原因仅仅是苹果生产的一切都是那一目标的反映。

为什么大多数公司不曾拥有追随者

问题在于，大多数公司领导人都没有真正明白什么能带来真正的长期增长。多数公司看到了短期的销售收益、增长和不错的产品特性，于是就想：“哇，只要坚持做这些，我的公司就能持续增长，最终便有了强大的用户基础。”这么做短期内或许有效，可如果要开发忠实的客户群，这绝非长久之计。事实上，这只会带来相反的效

果。销量和增长一旦到了终点，你便会看到用户不再购买公司的产品，甚至更糟糕，用户会因为产品突然涨价而对你的公司心怀怨念。

要进一步理解这一观念，最简单的办法是考查其他一些把用户转变为布道者的公司。

■ **Y-Combinator**：Paul Graham所做的任何一件事都是个人信念的反映。他的公司围绕着如何帮助初创公司成功开展业务。他的博客同样就如何帮助初创公司这一问题提供建议。天哪，就连他的妻子也是创业社区的提倡者。Paul Graham就这样食用着、呼吸着、魂牵梦绕着自己帮助初始创公司去改变世界的信念。创造何种产品无关紧要，重要的是**他的目标始终如一**。假如明天Y-Combinator发行一张关于帮助初创公司扩张的DVD，你觉得结果会怎样呢？成千上万的初创公司老板会毫不犹豫地购买它。为什么？因为他们**信任**Paul Graham，相信他的初衷必是极好的。Paul的所作所为已经一次又一次地告诉我们：他的生活就是这一信条的反映。他所做的一切都与“帮助初创公司”这一信念保持一致。



这是否意味着Paul制作的DVD与世上其他DVD相比含有更多对初创公司有价值的信息呢？非也。但对初创公司的老板来说，这并不重要。他们信任Paul，相信他总有最好的出发点。因此，即使其他人打算制作一张最美妙的DVD来帮助初创公司成长，人们却依然会推荐Paul的，而不会推荐其他人的，**因为他们明白Paul代表什么**。

■ **Quora**：我想说我本人就是个Quora的布道者。每当有人问我“如何做到X”或者“当Y发生时我该怎样？”我几乎总会说：“上Quora看看。”实际上，即使别人没有问我问题，我也会让他们去看看Quora，从而收获一些有价值的见解。这是否说明Quora对一切可能的问题都能给出最佳答案呢？完全不是。但对我来说这不重要。我**信任**Quora，因为这个公司做的任何一件事都**与其信念保持一致**。Quora会强制折叠虚假的答案，通过严格的制度限制可以提问的问题种类，他们甚至有版主在不时地编辑用户提问。这一切都是同一目标的不同表现，而这一目标就是为用户带来最具智慧的探讨和回复。有时，看到版主修改了自己的问题会让人不快，但公司这么做是为了与“创建具有最智慧信息的竞技场”这一大目标保持一致。

尽管不在Quora上班，我也百分百地确信Adam D'Angelo和Charlie Cheever会不断向团队重申公司的目标是什么。那是公司的出发点。你首先需要向团队传达公司努力的目标是什么。这样，即使你不在场，员工也清楚该做何种决策。我相信，Quora的版主会看着一个问题说：“伙计，Adam是不会允许问这种问题的。”这正如苹果的设计师会看着一款产品说：“伙计，Steve是不会允许做这种设计的。”

■ **Amazon**：这家公司拥有数百万计的布道者（我也是其中之一）。他们是如何做到的呢？公司成立初期，人们曾严厉地批评Jeff Bezos允许用户对图书做负面评论。人们告诉Bezos，允许这种负面反馈出现在网站上简直是疯了。评论家说，如果他允许用户对图书留下不良的评论，那就不会有人愿意从Amazon买任何东西。然而，对

Bezos来说这没什么，因为他在乎的不是如何用最简单的方法获取最大销量；**他在乎的是如何提供准确而诚实的细节信息，从而帮助客户做最可靠的决策**。猜一猜结果怎样？用户慢慢开始信任Amazon。用户渐渐意识到，Bezos最能维护他们的利益；他不是那种哄着骗着做订单的钞票饥渴型CEO。这种信任对Amazon至关重要，它使Amazon之后推出的各种服务（EC2、Mechanical Turk、Payment等）同样能够成功。他们之所以能在这么多产品上胜出，正是因为用户一直信任他们，相信这家公司仍然代表他们的最大利益。Amazon实现的所有产品特性，无一不体现了公司创建流畅购物体验的初衷（推荐引擎、Amazon Prime、诚信评价、1C等）。

我上Amazon网购的历史已经有8年之久，从没遇到延误或丢失包裹的情况。低廉的价格，出色的客户服务使得Amazon成为许多人网购的首选平台。运营方面，Bezos也是无可挑剔的。无论打算实现何种产品特性，他都要确保“让客户受益”这一理念处于每一种新服务的核心位置。我信任Amazon胜过信任世界上任何其他公司，而这只因为Bezos一次又一次地证明，他唯一在乎的就是提供最安全、最简单的在线购物体验。

如何将用户变成布道者

乔布斯、Adam D'Angelo和Jeff Bezos都是远见卓识之人，但他们不能仅凭忠实的客户来打造一家成功的公司。要把客户变成布道者，首先要把员工变成布道者。

如果你正在创立一家公司，可以考虑从以下步骤入手。

- 确立公司的目标（如果目标是尽可能多地赚钱，那么客户永远都不会变成布道者）。公司为之拼搏的缘由是什么？制定一条用户可以加入的理由。
- 食用你的目标，呼吸你的目标，魂牵梦萦着你的目标。你创造的每款产品，做出的每个决定，提供的每项服务都要与支撑公司的目标保持一致。不论你是在撰写一篇博客还是在创造一种电子产品，目标都应始终如一。
- 不断向所有人传达你的理念。包括合伙人、员工和用户。让大家都清楚你代表的是什么，公司的大图景是什么。CEO的生活应成为公司目标的活动符号。
- 永远不要背叛你的员工和客户。你必须以身作则。一旦失去信任，想重新得到便难上加难（如果还可能的话）。Groupon和LivingSocial两家公司提供的服务基本相同，为什么Groupon招致的怨气却比LivingSocial严重得多？那是由于不再有人相信Groupon能实际帮助小企业，因为它的行动与目标背道而驰了（雇用极善言辞的推销员去骚扰小企业，生意一旦谈成后面便是糟糕的客户服务，从IPO中捞回扣……）。

变用户为布道者并非易事。通常这需要成年累月的长期正确决策，然后用户才会敞开心扉并完全信任一家公司。随着创业公司的成长，团队成员不断增加，坚持公司初创时设立的价值观念将越来越难。然而，一切从领导开始。公司领导人必须过一种与公司目标一致的生活，做出的决策也必须完全符合公司的目标。在这之后，公司领导人还必须将公司为之奋斗的目标不断传达给所有的人。

如果你的生活能成为公司目标的符号，同时你不断向合伙人和员工重申自己为之奋斗的目标，那么，即使你不在场，团队同样能做出正确决策。藉此一家公司可以成长至数千人而秉持始终如一的价值。藉此苹果的员工可以做出上层决策而不需要乔布斯一步步手把手地指导。藉此客户便能渐渐加入你的征程，开始宣传你的目标。藉此你便可以将用户转变为布道者。



Rohan Jain

2010年获得加州大学伯克利分校经济学学士学位，在数据分析、风险管理和销售领域拥有丰富经验，他希望能帮助创业公司开发出用户钟爱的产品。目前他是移动支付公司Square的风险分析师。

产品经理与项目经理的关系

文 / 陈雄

互联网产品经理是一个很有意思的职位，与传统行业的产品经理有很多不同，虽然职位名称以“经理”结尾，但却是无授权领导，只对产品负责。从职级关系上看，产品经理和软件研发工程师是同等职级的。除了产品经理外，互联网产品团队中还存在着另外一个“经理”——项目经理，项目经理要管理产品，产品经理要确认项目，他们之间的关系是怎样的？同为“经理”，产品经理和项目经理工作中会出现什么冲突，怎样解决？本文将结合自己的工作经历对产品经理与项目经理的关系展开来讲一讲，希望对有志从事产品经理这行和已经入行的新人有所帮助。

产品经理与项目经理的异同

PM，在互联网公司我们经常这样称呼产品经理（Product Manager），而项目经理也叫PM（Project Manager），产品经理对产品管理负责，项目经理对项目管理负责。用几个关键词来描述这两个角色的特征，产品经理关键词：产品、需求、用户，项目经理关键词：项目、排期、人月。具体关系从以下几个方面来讲。

■ 从专业出身来说，产品经理在国内大学并没有对应的专业。这是一个对专业没有特殊要求的职位，在我的产品同事里有学心理学的、学英语的、学工业设计的等。产品经理的教主，百度前产品副总裁俞军，他是学化学出身的。虽然入行看似没有门槛，但成为一名合格的产品经理是需要经过一番磨练和积淀的。而项目经理基本是计算机、软件工程等专业学科出身。在工作中，项目经理需要在掌握技术的基础上，对开发资源进行分配管理。

■ 从知识领域来说，产品经理的知识非常广泛，对产品发展需要有战略规划能力；挖掘用户需求时要求数据统计和调查分析能力；产出交互设计和UI设计时要求有一定的工业设计与美术功底；和技术人员PK时，更需要懂得基本技术知识；产品上线运营后，需要有市场营销的知识作为指导。同时，产品经理是老板和用户之间的沟通协调者，这需要产品经理具有管理老板需求和用户需求的软能力。而项目经理必须懂技术，他通常由团队中项目经验很丰富、技术能力很优秀的RD担当，深厚的技术功底和牢靠的项目管理知识是项目经理的必备生存技能。

■ 职责不同。产品经理负责一个产品从idea产生到最终将产品放到用户手中的整个产品周期；而项目经理的职责是将产品细化为可量化可实现的一个个项目计划，并监督和控制计划保质保量地执行，只负责执行阶段。从某种程度上说，产品经理的职责没有终点，产品从idea诞生的那一刻起，产品经理的职责就开始了，只要产品存在一天，产品经理也在岗一天。项目经理的职责和项目的特性一样，有始有终。他可以完成一个项目之后再无缝切换到另外一个项目中去，但产品经理不能，他必须持续不断地负责一个产品并和这个产品一起长大。一个产品的成长会伴随着无数的项目更迭，在我负责产品的多个版本迭代中，就和多个项目经理合作过。

用一个比喻可以更深入地理解两个角色的不同，假如把产品比作一个小孩，那么产品经理就是这个小孩的妈妈，妈妈从怀上宝宝前就开始规划宝宝的一生，从怀胎十月，到呱呱落地……妈妈都要为自己的孩子负责。而项目经理是孩子成长过程中所遇到的老师。老师可以教不同的孩子，从一年级培养到六年级，孩子毕业，老师又开始教另一批孩子。工作中项目经理可以无缝切换，但

产品经理不能，就如小孩的老师可以换，但妈妈却不能随便换。

最后比较一下产品经理和项目经理的领导力。产品经理的领导方式是无授权领导，即产品经理管不了人，只管事儿。产品经理靠完善的流程和优秀的沟通能力完成整个团队的无授权领导。在平常工作中，我需要用30%的时间与团队沟通，因为产品经理既是团队的润滑剂，又是凝聚团队的万能胶。在产品内外出现任何问题时，第一个站出来的，永远是产品经理。项目经理领导力体现在技术团队的领导上，他具有调兵遣将的实权，他带领技术团队实施项目，他是技术团队的精神领袖。项目成员背景资历各不相同，项目经理作为项目领导人，要将他们组织起来并合理有序地为着一个共同的目标而努力。

产品经理与项目经理的冲突解决

百度、腾讯这样的大型互联网公司对产品经理和项目经理做了严格划分，但国内很多中小型互联网公司的实际情况是这俩职位由同一人担当，由于中小公司人力成本的限制，这可能是更符合实际情况，是更有效的一种组织方式。

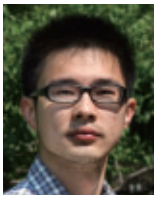
在工作中，产品经理也会和项目经理产生很多矛盾。最常见的是产品经理会在项目执行阶段提出需求变更，项目经理由于时间和人力限制，通常会拒绝产品经理的需求变更。产品经理为了更完美的满足用户需求，会对交互和功能不断地优化和变更，持续改进做加法；而项目经理为了确保在既有的资源内按期按量完成项目，并降低因变更带来的风险，会控制需求做减法。两者如何平衡其影响范围，举一个工作中的例子。

一次产品迭代升级，一个新功能点屡次被不同手机机型的用户提出要求加上，但由于开发资源限制，需求评审时考虑放到下个版本做。随后在开发过程中，我收到来自论坛、贴吧、邮件各个渠道的用户反馈越来越多，均要求加上这个功能，这些用户都是我们的核心用户。如此具有用户痛点的功能如果放到下期再做，可以想象，产品上线后，我们可爱的用户发现新版本居然没有这个功能，对我们会是多么的失望。于是我决定提出加上这个功能，经过产品组内部评审后，我向项目经理提出了需求变更。项目经理以项目进行中人力不足、需求变更风险过高为由拒绝了我。

这时，我把我们软件装到测试手机上，假定了一个常见的用户使用场景，让项目经理把自己当成用户去使用。经过一番折腾后，项目经理告诉我：“没有这个功能，的确让我用得很不爽。可是我现在没人分配给你去做这个功能啊。”接着，我又把后台关于这个功能相关的统计数据给项目经理看了一些，这次他在一定程度上同意了这个需求变更。由于存在开发资源不足的问题，接下来，我和项目经理一起找leader解决，由于这个功能点较独立，因此leader加人解决了这个问题。最后，项目按期结项，用户在新版本中看到了这个功能，都说帮了他们的忙。

这个冲突解决，表面看似是项目经理向产品经理妥协了，其实是项目经理和产品经理向用户需求妥协了。一切冲突的解决都需有理有据，晓之以理，动之以情，站在用户的角度去看问题。这个矛盾的解决实际是一个双方博弈的过程，最后会以一个双方与团队都能接受的结果确定下来。对于产品经理提出的需求变更，如果自己都拿不出能说服团队的充分理由的话，那么这个需求变更很大程度上会被项目经理PK下来。

互联网产品开发中，很少有使用传统软件开发的瀑布模型的情况，快速迭代、用户导向、不断试错是其鲜明的特征，因此需求变更在互联网产品中显得更为频繁。产品经理是用户的代表，与项目经理博弈的过程不可避免，解决之道是寻求一个双方的平衡点，以满足用户的需求为底限。一个优秀的产品经理不仅懂得把握用户需求，更能清晰地知道项目延期带来的损失。一个优秀的项目经理不仅深谙项目控制的意义，而且知道产品的市场定位和给用户带来的价值。产品经理和项目经理在团队中同等重要，缺一不可。



陈雄

前百度和腾讯移动互联网应用产品经理，北京大学硕士。移动互联网应用发烧友，主要从事LBS、SoLoMo和WIN8等领域的产品工作。

做不了，谁的错

文 / 邱岳

不论是多么优秀的产品经理设计的多么伟大的产品，若最终无法落地成型并递送到用户手里接受市场的检验，都是纸上谈兵，毫无意义。而工程师恰是这最重要的一步中最关键的角色。在把蓝图变为最终产品的过程中，产品经理最担心的就是看到工程师摇摇头，说出类似“数据量太大，同步做不了”、“并发量太高，服务器撑不住，做不了”或者干脆就是简单的“做不了”这样的话来。遇到这种情况，产品经理该如何应对呢？本文从以下三个方面分析一些常见的场景，并根据经验给出可能的解决方案。

清楚的需求、价值和目标描述

众所周知，产品经理更靠近市场和用户，他们对产品的设计来源于对用户需求的把握和对市场前景的判断。在与工程师沟通的过程中，产品经理应该把产品设计的来龙去脉说清楚，而不只是交代产品功能需求，甚至给出方案。在这一方面，产品经理不妨问自己以下两个问题。

- **有没有说清楚项目的目标和价值？**如果产品经理没有对工程师讲述用户需求、开发产品的动机以及项目的商业目标，那么工程师很有可能没有透彻理解产品的价值或压根儿就不赞同产品创意，以一句“做不了”来搪塞也不意外。这背后的潜台词可能是“做这个有什么用”或“现在做这个就是浪费时间”。产品经理在任何时刻都必须与工程师紧密合作，把产品创意、开发产品的动机及对市场的判断等跟工程师交代清楚，让工程师理解为什么要做这样的事情（商业目标），以及为什么要现在做这样的事情（优先级）。这样做不但能减少一些“做不了”的回答，同时能让工程师对全景有清晰的理解，从而准确编码和设计功能，更符合产品经理的设计期望，满足用户需求。
- **描述的是实现方案还是原始需求？**有时，出于各种原因，产品经理可能会试图越俎代庖，不做需求描述而根据自己的设想直接描述“实现”。比如，产品经理提出要“一匹时速100公里的马”，工程师的回答自然是“做不到”。若产品经理将需求描述为“更快到达目的地”，工程师或许会告诉你，他掌握了一门叫“制造汽车”的技术，应该可以满足需求。不论是出于对工程师的不信任，还是过度膨胀的责任感，类似的情况都不可取。产品经理的职责是准确描述需求，虽然可以提出一些建议，但无论如何，设计和实现产品的任务还应该交给更专业的工程师来完成。

资源和限制

平心而论，这世上可能真没有什么绝对做不了的事，更多的是对“投入产出比”的权衡。下面举三个经常发生的场景。

- **“不值得做”=“做不了”。**工程师有时会把“投入产出比低”说成“做不了”，比如产品经理可能提出“在IE、Safari、Chrome、Opera和Firefox等几种浏览器的各版本中保证视觉、交互效果一致”这样令人发指的需求。虽说这并不是无法完成的，但为了类似的需求却要花费很大的代价。这时工程师所说的做不了，其实是对“投入产出比”的不认同。对于产品经理来说，这时应该首先主动解释清楚“产出”，即提出这样需求的目的，然后再跟工程师探讨“投入”，即要实现这一功能的成本。如果对成本有争执，可以试着咨询一下其他工程师，看看是不是有更好的解决方案可以降低实现成本。如果真的没有，那么产品经理要能准确权衡需求点的价值，从商业的角度判断“投入产出比”，对功能点进行取

舍。例如上面的浏览器兼容需求，产品经理不妨查看数据或做一下潜在用户分析，选几款市场占有率较高的浏览器进行支持。不过话说回来，产品经理很大一部分职责就是要根据各个功能需求的“投入产出比”来排定产品迭代节奏，这时产品经理应该敢于决策，不要被工程师对“投入产出比”的判断误导。此外，我建议产品经理能主动了解点技术，虽然不需要钻得太深，但要能大概判断一件事情的成本，或最起码在跟工程师沟通时能快速理解关键点就够了。

■ **历史遗留系统或既定的技术框架。**大部分工程师都喜欢盖房子，不愿意维护房子。因为在历史遗留系统上进行升级、维护，或者在既定的技术框架上开发产品时，难免会遇到很多技术限制。比如，我曾经提出个性化表单验证的一系列需求，但事实上我当时负责的系统使用的框架规范了所有表单校验过程，无法实现我要求的定制化需求。我觉得这种问题的处理方法与上面说到的“投入产出比之争”的处理方法类似。首先，产品经理与工程师一起找到对遗留系统或框架最熟悉的人，咨询是否有实现方案。如果没有很好的办法，转而评估“投入产出比”。若不做此产品需求，公司的明天岌岌可危，那么宁可重构系统或更换框架也在所不辞。但如果没那么严重，不妨想想别的办法。比如上面说到的个性化表单验证需求，就是在分析了时机和需求价值后，修改了交互流程，选择了一种框架支持的方式。

■ **被污染的系统。**还有一种可能，是由于系统代码被长期的“紧急”需求所污染而导致的“做不了”。这类需求通常是商业优先级极高的，面对这类需求，产品经理经常会要求工程师“越快上线越好，不用去管设计模式，也不用去管代码规范”。做互联网产品，“快”绝对没错，但必须要有节奏感，这些被类似的代码不断污染后的系统会留下许多陷阱，同时也会为未来的产品功能升级设置很多障碍，基于这些陷阱和障碍，很容易得到“做不了”这样的回答。我认为不论产品经理还是项目经理，一定要有意识地规避这样的风险，防微杜渐，确保工程师有足够的时间经常审查设计，并保证持续重构，使每次为了“快”而做出的设计牺牲得到及时的修正，否则不但未来的产品发展会受到影响，而且工程师也会因为总在救火而越来越疲惫，导致效率降低。

环境和人

除了上述的各种原因，还有一些原因是与“人”相关的，很多时候我们的环境让沟通很难有结果，比如下面提到的两种情况。

■ **局部利益之争。**如果你在大公司工作，这种事情很可能会发生，就是不同部门之间的边界、利益之争。有些需求，A、B部门互相推诿，都说“做不了”；而另一些需求，A、B两部门抢着做主要模块，都不想做周边功能，在评估系统实施时，各个部门可能会倾向于说这些周边功能他们“做不了”。这样的问题，我建议产品经理尽量不要在A和B之间过度沟通斡旋，防止由此产生的立场对立甚至矛盾升级，恐怕最好的办法就是去找系统架构师，从整个系统的角度给出专业的意见。

■ 还有要特别补充一点，如果你碰到的是胸无大志、只想混混日子的工程师对你说“做不了”，那最好的办法是不要浪费时间，直接换个人再问问。但如果对方是技术功底还不太扎实、经验不太丰富的工程师，那应该带着他一起去向更资深的工程师去请教，不但事关尊重，而且更是帮助工程师快速提高能力，尽快肩负起更大责任的好方法。

顺利的产品实现过程都差不多，做不了的情况却各有各的不同。“做不了”，并不是谁的“错”，我认为只是产品经理和工程师的配合问题。

当产品经理面临“做不了”的挑战时，应该主动与工程师沟通，从客户价值出发，一起合作解决问题，确保信息对称。只有这样才能保证高效地开发出满足客户需求的产品。



邱岳

阿里巴巴（中国）网络技术有限公司产品经理，从事开放业务和平台相关产品规划和设计工作。

如何寻找优秀的产品经理

文 / 沈烁

你需要一名产品经理？可能你还不是很清楚应该找个什么样的人，但能明确的是这个人一定能帮你做出并改进产品。也许市面上已经出现大量仿制品，你急需改进现有产品以脱颖而出；亦或你刚刚争取到一批非常不错的资源，希望有个人能把资源盘活；再或许你是一名资深的工程师，希望找个靠谱的人跟你一起把点子变成现实，一起做出下一个Instagram，被扎克伯格以10亿美元收购。

当然，对于一个没有做过产品但又想找到一个优秀产品经理的人来说，确实有些困难。就像我作为产品经理，不熟悉代码但又想找到有潜力的工程师一样难！那么，到底怎样才能雇用到一个靠谱的产品经理呢？

最简单的办法当然是去找已经做出过成功产品的人，但事实上他们往往都很抢手，无论你怎样绞尽脑汁向他们解释自己做的东西，告诉他们这也许将是下一个互联网热点，你的团队如何有活力且没有束缚，他可以放开手去干，但最后你还是竞争不过那些业内大鳄。

好吧，回到现实处境中，你迫于现状，只能从那些还未能做出成功产品来证明自己实力的产品经理中寻觅一支潜力股！至于潜力股他长什么样咱不好说，但一般都会具备以下一些特质。

有独立思考能力

我们每天打开手机和电脑都在接触大量的业内新闻和热门观点，例如某产品获得了上亿投资，创始人表示他们成功的原因是看重用户需求，重视产品设计，追求产品细节，每天都在快速迭代自己的产品等。这样的论述是不是看起来很眼熟？我相信很少有人会反对这些观点，这些话被反复传颂以至于说起来就像真理一样。但有没有人想过，为什么他们会选择这个需求，还有什么关联需求？为什么放弃其他的？用户需要数码产品，也需要买生活消耗品，但为什么数码产品的电商做得这么火？手机QQ也能发语音，为什么大家还都用微信？Evernote、Dropbox、Pinterest在国外如此知名，在国内会如何？

产品经理的首要任务是提出正确的问题，如果他只是默默接受别人的观点而没有任何问题，他也很难提出有新意的改进办法。如果你要将自己的产品全权交给面前这个人，可以问问他有什么改进方案，看看他是不是个爱提问题且有独立思考能力的人。

对需求敏感

广告界有这么一句话：卖的不是牛排，而是牛排的滋滋声。这是每个广告人和产品人都明白的道理，消费者买的不是产品本身，而是产品带给消费者的感受。讲个我亲身经历的事吧，一天我将一瓶常温的可乐递给同事，他接过去失望地嘟囔一句“嗨，不是凉的啊”。我想，大家都有这样的感触。在燥热的夏天，常温可乐和一瓶冰镇可乐给用户带来的体验完全不同。虽然它们的价格一样，但价值却不一样，我们买的是可乐喝下去后那一声“嗝！哈~”的感受。

道理说起来容易，但我们经常犯的错误是拍脑

袋想出一个产品，然后沉溺于这个产品内部的结构如何精密复杂，如何在满足用户需求方面能自圆其说，而没有去认真体会用户细微和内在的需求。不能手里拿着锤子，看什么都像钉子。如果你想判断眼前这个产品经理是否对需求敏感，可以问问他所熟悉的产品都满足了什么需求，这样设计都有什么优缺点，设计的矛盾点在哪儿，如何平衡。太阳底下没有新鲜事，你找的可能不是第一个发现需求的人，但他一定是对用户需求敏感并且能提供最优解决方案的人。

另外，不同地域的用户需求也会有区别，我知道一位60岁的荷兰奶奶来中国游玩，每天都会用博客记录自己的所见所闻。而我的母亲每天只要能玩上连连看和浏览新闻就已经很满足了。对需求敏感的产品经理除了看国外有什么新的模式，更会关注自己身边的人真正需要什么。

好的产品经理不仅有个好鼻子，能闻到用户需要什么，还要知道满足需求的时机是否成熟。电子商务的蓬勃发展离不开物流行业的崛起、支付的安全性、网民网购意识的培养，现在建立一个电商网站和十年前建立电商网站的发展速度有天壤之别。由于电商的兴起，建立电商相关产业链也成为了可能，如现在社会化电商和商品推荐算法的兴起。这些紧随市场需求的产品才有机会快速成长。

对市场敏感

在IT圈打拼的人都知道，互联网产品满足了需求才有用户，但有用户不一定有市场。互联网的基因就是开放与分享，我们每天使用的很多互联网产品都是免费的，从没想过聊天看新闻看视频还要交钱。但对于身处这个行业中的人来说，做产品的目的可不是为了公益，就算是公益事业也需要最基本的开销吧，租办公室、聘工程师甚至水电都是开销，最终我们还是希望产品能够带来可观利润而不只是做个广受追捧的免费玩具。

因此，要么产品本身能产生销售额，像搜索、游戏、广告；要么它能提供增值服务，像虚拟形象、空间装饰；要么它能为其他业务提供支持，像电商、支付、数据分析；还有就是做其他产品



产品经理潜力股的特质

的上游，像浏览器、导航网站。产品向市场输送了价值，市场才会给予回馈。产品经理都有满足用户需求的意识，他们深知要去了解用户需要什么，但在IT行业中用户往往不是客户，在保护用户利益的同时也要兼顾客户的利益。当面前这位产品经理对某一款产品说得兴高采烈时，你一定要问问他这款产品都为谁带来了价值、提供了哪些服务或者支持，以及它的上游和下游都是谁。

懂技术、懂设计

福特先生有一句经典的话“如果我问我的用户，他们只会说要一匹更快的马。”的确，如果我没见过汽车飞奔，没亲自驾驶汽车体验过汽车的便捷安全，我的回答也是一匹更快的马。用户只是会根据现有的体验提出期望，就像我在玩电子游戏机之前只会想纸牌有没有更吸引人的玩法。汽车刚被发明出来时是用蒸汽驱动的，用做城市间的运输。由于它结构复杂经常损坏，而且价格昂贵，一般阶层的老百姓根本承担不起。经过发明内燃机，福特先生发明流水线并实现量产后，汽车的性能才趋于稳定，价格也降了下来，真正懂

技术的人给出了解决方案。

可以想象，如果我穿越到了那个年代，跟造车的人说我知道未来的趋势，我们需要制造内燃机建立流水线，造车工人围着问我该怎么做，我肯定呆住了，我只能画出那个饼，但完全不知道技术上怎么实现。不过换做福特先生，虽然他当时还不知道未来的趋势，可一旦他认识到了，他一定知道怎么做。但只懂技术还不够，借用一位前辈说的话——产品经理可以不懂技术，但要懂技术能做什么。

另外，产品经理还要懂设计。这里说的设计不是简单意义上的会不会做图，能不能用Axure做出页面和交互效果，而是指信息设计。有个笑话是说Facebook登陆中国，应国内用户要求开发了注销账户功能，但过了没多久，又有用户提出要求恢复已注销账户的功能，又过了一段时间，又有人提出永久注销账户不可能恢复功能……虽然是个笑话，但说明用户虽然在需求方面是权威，但用户不懂设计，你需要追问他想这么做的目的，而不是轻易采纳他的修改建议。

设计是一种权衡，它代表的是产品的气质，而不应该是使用者的。如果你想考验产品经理的设计功底，可以拿他设计的一个页面来问用户在什么情况下会到这个页面来，你想给用户看什么，为什么这样设计，以考验他是否明白产品的整体定位对于一个按钮的位置有什么影响，以及考察他是否有让页面简洁明了传达信息的意识。

整合资源能力

罗马不是一天建成的，凯撒大帝不是一人打的天下，乔布斯也是借助乔纳森和库克的力量才重振苹果帝国。我有好几个朋友都表示过想开淘宝店，但问及为什么没有开时，他们有的表示没有货源，有的说不会设计。

而产品经理遇到的问题也不会比开个网店少，有的产品经理擅长使用有限资源开发出小而精的产品以获得口碑传播，有的则擅长左右逢缘通过人去解决问题。是使用有限资源四两拨千斤还是凭借三寸不烂之舌争取更多资源就看你的本事了。

以上指的都是优秀产品经理的潜质，即软能力，除此还有硬能力，虽然没有软能力那么重要，但如果你要雇用的这位产品经理能够具备，尤其在做一些特殊产品时，会很有帮助。

编程能力——如果他会写代码，他就会知道电脑能做什么，擅长什么。像安全、支付这样重度依赖技术的产品会非常需要懂编程的产品经理。

数据分析——数据可以是某个重要的业务指标，也可以是用户留在网上的足迹，重要的是他能从中发现规律，为下次改进产品提供理论依据。像广告、电商会更强调产品经理的数据分析能力。

文档与原型——文档原型是产品经理的入门功课，写得工整会让看的人心情愉悦。特别是在大型公司，工程师、设计师都需要通过文档和原型解读产品经理的想法。

如果你真的找到一名具备以上潜质和能力的产品经理，祝贺你。不过，雇用之后，如何调动他们的积极性，也是一个重要问题。简单来说，实际上产品经理不需要严加管理，只要你和他在产品的方向上达成共识，给他足够大的发挥空间，给予他足够的信任，放手让他去干吧。我知道你开始肯定会有些担心，就像把心爱的古董放到一个孩子手里，生怕他给摔碎一样。我们从小都有这样的经验，那些父母说了不让做的事，到现在也不能体会其中的原由。反而是那些让我们吃了苦头的错误，才能亲身感悟到其中道理。对于产品经理也是如此，在他做错时让他认识到错误的严重性，与他一起快速回到跑道上才是上策。总之，请一定要信任他，你能带给他的最大帮助就是使他专注，他一定会认真对待你们的产品。



沈烁

曾参与电商及社区产品设计，现从事社会化电商方面工作，是一名5岁的产品经理。

技术人员如何转行做产品经理

文 / 苏杰

经常看到这样的问题：

“产品经理都是招有几年工作经验的，我原来做技术，要转行怎么做？”
“研发转产品，真的很痛苦！”
“我是软件产品项目经理，如何成功转型做产品经理？”
……

所以，我们来聊聊技术人员转行做产品经理的话题，分三步。

首先，问问自己真的想成为这样的人吗？然后，在出手之前必须找准自己的位置，并且做出一些努力。最后，谈谈转型之后应该注意哪些。

确认自己真的想做产品经理

既然你在看这篇文字，我就默认你已经对产品经理这个职位有些了解，并且“雾里看花”地身不能至，心向往之。比如那种指点江山的气质、实现想法的成就感、动动嘴皮子就一大帮人帮忙干活……当然，也可能你是觉得年纪大了，技术做不下去了……所以，关于她的好，我就不再说了。在跳入产品经理这个“火坑”之前，有必要说一下外人看不到的一面。

你会被“强奸”，比如是很强势的运营同学，被他们逼着做自己不认可的需求；
你会在产品上线后，被无数用户劈头盖脸地骂“这是哪个SB做的”，而你就是那个SB；
你会无奈地回滚，发现自己没想到的一个小细节，导致产品只能回到几个月前的版本；
你会被老板逼着2周做出来，然后被开发告知至少要2个月；
你会突然得知项目取消，然后老板要你把这个消息告诉那帮天天加班到12点的兄弟；
你会被扔给一个烂尾楼产品，除了你以外，大家都没有任何士气；
你会发现老板和老板的老板观点不一致，最终的方案却要你来决定；
你会和其他产品经理吵架，老板怪笑着表扬你，说这是对工作有要求；
你会通宵发布，然后拉上同学们一起聚餐——是早餐。

所以，你还想做产品经理吗？如果想，请继续往下看。

意愿没有问题了，还需要能力，这里的能力是指一种个性特质。做产品的大前提是要喜欢做产品，不然将来你痛苦，团队痛苦，用户也痛苦。网络上那么多好玩的应用，是很有意思，通常自己想做产品的同学都会去大量尝试、注册各种各样的产品，去用，去玩，去想……如果你没有这个特点，那就真要好好想想了。

尤其需要进一步明确的是：你到底是喜欢做用户，还是喜欢做产品经理？

如何判断这一点呢？可以这样做：

当你对一个产品感兴趣时，回想一下脑中萦绕的问题是站在用户的角度，还是站在产品经理的角度。通常，用户会去想怎么用这个产品，才能带给自己更大的好处，产生更大的效用；而产品经理则习惯于绕过表象，从背后看问题的本质，思考怎么设计这个产品才能更好地平衡用户目标与商业目标。

这么说还是太抽象了，举几个例子。

你在看电视时，是把自己当观众，还是会不由自主地去想导演为什么要设计这个环节？
你在上网时，碰到各种不爽的地方，只是骂两句娘，还是会去思考怎么改进？
你在老公/老婆要买新衣物时，是直接和Ta讨论绿色好还是蓝色好，还是会忍不住问——你为什么要买这件衣服？是在什么场合穿的？
你在添置大件时，比如买房买车，是看完各种资



讯后模糊评估，还是会自己做一份文档，建个模型仔细量化比较？
你在工作时，只是被动地接受产品经理的需求，还是会主动思考，和产品经理一起讨论，并且你的意见不时被采纳？

如果看到这里，你发现自己更多地是站在产品经理的角度想问题，一想到这些问题就热血沸腾，欲罢不能，OK，整个行业都需要你！但是——

找准位置，做出努力

已经很长一段时间了，不断收到邮件，有应届生、技术人员、运营人员等，说因为种种原因，发现自己真的很喜欢做产品经理（意愿），并且感到自己无比的适合（能力），好像这个职位就是为自己设计的云云，但！却屡屡应聘失败，问我应该怎么找到一个产品经理的岗位。

每次我都不知道怎么回信，因为每个人、每个公司的情况都不一样，这个问题根本没有标准答案，不过次数多了，后来多少总结出了一个方法，很简单，给出统一回复——

Hi，感谢对我的信任，你自己先要想清楚一个问题，如果有选择，一家公司为什么不选择一名有经验的产品人员，而选择你，期待你的答案 :)

让我失望的是，一般这个问题抛过去，有一大半的人都没有回应了。

首先，可能对方会觉得我是在敷衍，或者挑衅，其实我问得很实在。大家站在招人的角度想想，通常名额有限，我放弃了一个有经验的产品经理而要你，你总得给我一个“难以拒绝的理由”吧，就像《教父》里的经典台词——“我将给他一个他无法拒绝的理由”。然后，我才可以拿着你的理由去把简历转给合适的人，这是在帮你争取机会，对吗？

其次，在产品经理的工作中，经常会碰到各种有挑战的问题，如果你被一个简单的问题就挡住了，只能说明，要么你性格上不适合做产品经理，要么对自己的目标不清楚而导致没把答案想透，要么你根本没有什么“特别的理由”，对这个职位并无深入了解，只是一时兴起，这些对产品经理都是很致命的弱点。

因此，如果你真的想做产品经理，希望先把这个问题回答了，而答案，应该是和你过去做过的事情有关的，如果换了一个人也适用，那就等于没用。举个例子，比如下面这段，其实就等于没回答：

虽然自己只做过传统行业的传统型产品工程师，也只负责传统行业的营销工作，您在书中说到，人人都是产品经理，不管以前是做什么的，都可以转行做产品经理，我相信自己也是人人中的一员，在互联网蓬勃发展的今天，可以亲身从事这个行业的工作。
作为一个转行工作的人，自己是做好了以新人的心态开始工作，而且也会切合实际地选择一些我自己能够胜任的工作，您在书中也描述过几个可能入行的切入点，偏商业的可以先做“运营专员”，对产品做推广策划的工作。

我只是告诉你一种可能性，但不会因为你认同这个可能性，就能找到一份工作。我们不想知道“你是怎么想的”，而想看到“你做了什么”。其实道理你早就懂了，就像追女孩，你只是整天意淫她，是不会成为她接受你的理由的。

Just Do It，很重要。

你得充分利用已有的知识结构、资源，找到一条

最近的入行之路。也许你要失望，因为真的没有捷径，真的没有银弹！对于想转行的技术人员，我的建议是先在本职工作中找到与产品有关的事情做一些尝试，并且考虑先从产品经理周边的职位做起。好在产品经理的职责范围什么都涉及，这种事情总是能找到的。

比如你是做开发的，那就经常要参与需求评审，不妨比别人多用点功，每次都预先了解需求，多多思考，然后在评审会上对需求提出自己的合理建议，时间长了大家都会觉得你很有想法，做产品也许不错。关于职位，可以从“需求分析师”切入，这有些像系统分析的工作，比如业务逻辑、流程图，都是你已经很熟悉的，你可以在这个过程中慢慢培养商业的感觉，重点体会某个产品功能是为了满足商业上的哪个需求而做的。

你还可以利用业余时间做很多事情，比如一位同行举的例子——

每天看20条以上的互联网新闻，并订阅三五个大更新量的互联网博客，逐条细看一个月。随后再订阅二三十个博客，有选择地精读，常年坚持风雨无阻。

浏览新闻一个月后，选择一个自己主攻的细分产品类型，比如空间、微博、问答、点评、团购、LBS等，找到该领域国内的一两个领先者，国外的一两个领先者，泡进去玩。必须以纯粹的用户角色去泡，去交很多朋友。

对于自己选择的产品类型，建一张Excel表格，找出十来个关注对象，把它们的产品点拆分开来，记录你的研究心得与版本动态。每周更新一次。

这样坚持做下来三个月，对于能不能转产品，该怎么转，自己心里大致也有个数了。哪有什么良师，掌握好的自学方法才是王道。

发现没有？其实当你能很轻松地回答“统一回复”时，你已经做了很多为产品经理打基础的事情，我相信，机会离你绝对不远了。

转行之后，一些注意点

技术人员转型做产品经理有其无与伦比的优势——对可行性的敏锐嗅觉，但这一点，用不好也会成为劣势——过早地陷入技术细节的思考。下面是享有世界声誉的产品管理专家Marty Cagan在《启示录：打造用户喜爱的产品》中提到的几点建议。

第一，要清楚地意识到自己和目标客户是截然不同的。这需要花一些时间和真正的客户交往。

第二，学会“移情”（empathy），懂得站在用户的角度思考问题。花时间与用户做面对面的沟通，但要注意这并不意味着用户能提出真正的产品需求；挖掘产品需求是产品管理的任务。

第三，转变思想。这个问题看起来简单，但当你面临一个两难抉择，比如项目发布时间与用户体验出现冲突时，你才能体会其中的困难。

第四，保持谦逊的品格。仔细倾听来自用户的声音，日复一日，你的理解力会得到极大提升。但前提条件是必须拥有开放的心态来面对用户的批评。

第五，改变讨论风格。产品定义和用户体验的决策中一般不存在什么标准。这时就需要你提高说服和辩论的技巧，让他人接受你的观点。

第六，处理好和原部门的关系。需要学会放手，让工程团队做好他们的本职工作，并参与到产品开发的流程中来。产品管理的工作已经够让人头大了，相关的技术决策就放手让他们来处理吧。

Marty建议，各位的公司应该建立畅通的渠道为开发人员的转型提供便利，我也认同，这样一定会培养出许多杰出的产品经理。即使转型不成功，但在这过程中，大家学到产品经理的做事思路和思维方法，也会对今后的工作、生活都大有益处。最后，还是用我在《人人都是产品经理》中的一段话来结束——

不是每个人都能以产品经理为业，但在我看来，产品经理是一类人，他的做事思路与方法可以解决很多实际的生活问题。只要你能够发现问题并描述清楚，转化为一个需求，进而转化为一个任务，争取到支持，发动起一批人，将这个任务完成，并持续不断以主人翁的心态去跟踪、维护这个产物，那么，你就是产品经理。至少，你已经是自己的产品经理，这才是“人人都是产品经理”的真谛。



苏杰

阿里巴巴集团产品经理，负责集团内部产品经理的成长，关注用户研究、需求分析、产品规划、数据分析、知识管理等领域，特别是电子商务行业。

产品经理在市场调研中扮演的角色

文 / 李继斌

在IT行业中，提案开发新产品或者改版老产品，都或多或少地需要做市场调研，其重要性不必多言。市面上，关于市场调研方法论的图书或资料有很多，本文在这方面不做赘述，有兴趣的读者可自行查看。本文重点探讨的是，作为产品的管理者，产品经理在产品的市场调研过程中，应该扮演什么角色？

在回答这个问题之前，我们先来简单梳理一下，产品的市场调研过程中都包含哪些步骤，如图1所示。



图1 产品的市场调研过程

不同产品在不同场景下的市场调研过程会略有差别，但大体上都是这个过程。下面我们来简单对图1做一个说明。

- 明确问题：明确需要做市场调研的问题或需求。
- 制定方案：制定市场调研的方案和实施计划。
- 实施调研：分阶段、有步骤地实施调研计划。
- 分析数据：统计、整理和分析市场调研数据。
- 得出结论：对比市场调研前提出的问题，得出调研结论，制定下一步的行动策略。

市场调研不只需要产品经理，还涉及到企业内部的众多部门和其他角色。事实上，产品经理大可不必、也不可能事事躬亲。无论从市场调研本身的专业性上讲，还是市场调研工作的时间性上讲，产品经理都无法独立完成。

回到本文开头提到的问题：产品经理在市场调研过程中，究竟应该扮演什么角色呢？

我认为，产品经理只需要把握好以下几个关键因素即可最大限度地保证市场调研的成功。

- 找到值得解决的市场问题。
- 确保市场调研的可行性。
- 对比问题，产出调研结论和行动策略。

找到值得解决的市场问题

通常来说，IT行业中，产品经理提出市场调研的需求，主要来源于以下两个方面。

- 产品经理所处的行业或领域中出现新的需求；产品经理有了新的想法，想要满足这些需求，需要调研该想法的可行性。
- 产品经理负责的现有产品收到了用户的意见反馈或者受到了其他竞争产品的冲击，需要改版现有的产品，调研改版的方向和目标。

这里，以现阶段互联网行业中较为流行的社会化导购网站为例，针对上述的第一个市场调研需求，我们模拟以下场景。

假如你是一个处于电子商务领域的产品经理，你或许会看到以下需求。

经过这几年的高速发展，国内的电子商务巨头已经解决了“商品信息、信任支付、物流配送”等基本问题，用户也已养成了“网上购物”的习惯。但新的需求又出现了：用户在琳琅满目的海量商品面前，很难找到自己想买的，同时也不知道自己该去哪里买。

这时，你或许就会有了新的想法：怎样设计一个



产品来解决用户“买什么、在哪里买”的问题。

如果你看到了这个需求，并产生了这样的想法，那其实已经找到了一个初步的市场问题，但这对市场调研来说是远远不够的，你需要针对这一问题做如下探索。

■ 新产品应该选择哪些商品类目来做导购?

不是每种商品类目都有“买什么、在哪里买”的强烈需求，如标类产品（3C数码类等），用户往往会有很明确的购买目的以及购买场所的选择。但非标类产品（服装类等）就不一样了，用户往往需要大量的搜索和比较。对于女装来说，女性用户甚至要跑到线下专卖店试穿以后，才会决定“买什么”。

这时，就需要调研新产品究竟要做成“专注于某一商品类目的导购”，还是做成“全商品类目的导购”。换句话说就是，要做成“美丽说”，还是要做成“堆糖”。

当然，类目的选择不仅是要考虑用户需求，还需要考虑“某一类目的市场规模”等因素。

■ 产品要做成什么形态?

选择了类目之后，下一个需要解决的问题就是，产品形态是什么样的，是要做成UGC（User-Generated Content，用户创造内容）模式，还是要做成EGC（Editor-Generated Content，用户创造内容）模式。换句话说就是，要做成“蘑菇街”，还是要做成“LC风格网”。

这一点很重要，因为模式的选择决定了产品的形态，也决定了这个新产品将来是要投入更多的技术人员（需要有更精准的算法模型来筛选用户分享的内容），还是要投入更多的编辑人员（需要不断地编辑创作出新的内容来留住老用户并吸引新用户）。

以上场景只提出了一部分问题，想要说明的就是，在做市场调研前，产品经理一定要通过多种方法找到值得解决的市场问题，并且将问题尽可能细化到最小粒度。

确保市场调研的可行性

找到值得解决的市场问题之后，紧接着需要做的就是制定市场调研方案和计划，并开始执行。这个过程一般会由市场部门或用研部门，甚至专业的第三方公司参与进来共同完成。

在确定完市场调研方案和计划（包括具体的执行安排）后，有些产品经理会想当然地认为，自己的大部分工作已经完成了，只需要等待调研结果就行。如果你也是这样认为的，那么在调研完成

后往往会发现：有些问题没有被调研执行者覆盖到调研过程中，或者由于调研方式的原因，造成调研结果不是自己想要的。

因此，这个过程中，产品经理仍然需要时刻保持“对结果负责”的态度，把控整个市场调研过程，定期了解市场调研的进度，随时发现和解决出现的问题。

总的来说，这个过程需要注意以下几方面。

- 确保市场调研方案和计划能够完全覆盖到“值得解决的市场问题”。

产品经理不一定要独立完成整个市场调研过程，但一定要全程参与，充分对比调研前提出的问题，确保每个问题都有明确的结论，可用于指导下一步行为策略的制定。

- 调研过程中，根据前期的调研结果，做适当的分析，随时调整调研问题的侧重点。必要时，可以更换部分问题或新增问题。
- 和具体执行部门（市场团队、用研团队或者第三方公司）建立一个有效的沟通方案，确保具体的执行者足够了解方案，不会因为沟通问题，影响最终的市场调研结果。

我曾有幸参与过一个“平台满意度”的市场调研。调研的准备阶段是由平台的各个业务方自己提想要调研的问题，最后统一整理调研方案。但调研进行到三分之一时，我们发现有部分问题的设计不合理，影响调研用户的理解。还有一些问题，调研方案中没有提及，但被调研的用户却比较关心，反复提及。

基于这种情况，我们与用研团队和各个业务方一起修改了调研方案，更改了部分问题的提问方式和顺序，新增了在前期调研中，被多个调研用户反复提及的问题。

对比问题，产出调研结论和下一步行动策略

很多产品经理往往在“发现问题”和“调研过程”中做了大量的功课，但在收到市场调研的反馈数据后，却草草地分析得出结论，或盲目地依赖合作团队的分析结果。

当然，产品经理不一定要完全独立完成这个过程，但一定要全程参与，有自己的分析和思考，要充分对比调研前提出的问题，确保每个问题都有明确的结论，能用于指导下一步的行动策略的制定。

最后，为确保报告的可读性，撰写市场调研报告时，要把握以下两点原则。

- 尽可能多用图表，少用文字（文字要简明扼要、通俗易懂），文不如数，数不如表，表不如图。
- 清楚地标明行动策略和下一步的具体行动。

我之前做过一个“某业务增长动力”的分析报告，相信大多数产品经理对类似的分析报告都不陌生。

一般情况下，运营一段时间后，产品或业务数据指标都会有较大幅度的增长，找出正确的增长原因对下一步的运营方向会有较大的影响。

但我在这次分析过程中就犯了一个错误，仅对收集过来的数据做了分析，却没有对下一步的具体行动做清晰的说明，最终导致返工。

事实证明，当你分析完数据，真正思考下一步的具体行动时，往往会发现分析过程有些问题没有考虑到或者没有分析到，这一步的思考和之前的分析过程其实是一个相辅相成的关系，所以“思考下一步具体行动”的环节尤为重要。

本文讲述的是我的工作经验以及学习过程中的心得，供读者借鉴，帮助产品经理梳理清楚自己在市场调研过程中所需要扮演的角色，希望能对做市场调研的产品经理起到一定的帮助作用。



李继斌

淘宝花名方胜，天猫（淘宝商城）社区电子商务部产品经理，之前担任淘宝开放平台产品经理，负责过的产品有淘宝账号开放、开放平台OAuth2.0授权、淘宝合作网站电商化解决方案等。

数据启示的互联网产品设计

文 / 罗旭祥

Facebook产品设计经理Adam Mosseri曾经做了一个非常棒的演讲《Data informed, not data driven》，提到了“数据驱动”和“数据启示”两个概念。这里的数据驱动指一切产品设计行为都以数据为核心，而数据启示则指把数据作为辅助来影响产品设计。产品经理非常容易陷入数据中难以自拔。一方面是因为很多评估产品设计的指标通常是非常明确的指标数据，另一方面是因为产品经理通常对数据过于敏感。而Adam Mosseri的产品设计思想的核心在于不以数据驱动的方式设计产品，而是把数据置于合适的位置融入整体产品设计过程，既不过分依赖数据，又不完全脱离数据。我对此深以为然。Adam Mosseri认为过度依赖数据可能导致微优化（micro-optimization）和局部最大化（local maximization），所以数据应该是启示性的而不应该是驱动式的。那么如何充分利用好数据以协助产品设计呢?

用户行为是产品之魂，用户数据是产品之魄

通常，产品设计通过用户研究、概念设计、交互设计、细节设计等一步步地接近理论上想要的样子，但即便能认真遵循每一步骤细致地完成工作，产品距离用户想要的依然很远。原因很简单，我们所研究的用户角色的细分是有限的，而用户间的差异性是必定存在的。对用户分类的颗粒度越大离用户想要的越远，这是用户体验设计方法的天生缺陷。在常规的消费者数据挖掘的RFM（Recency最近一次消费，Frequency消费频率，Monetary消费金额）模型中，我们可以根据用户数据特征进行用户行为建模（利用xyz坐标建模），可得到用户行为分布图谱，如图1所示。

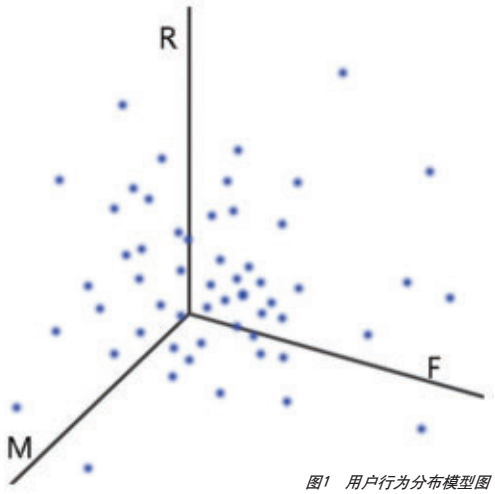


图1 用户行为分布模型图

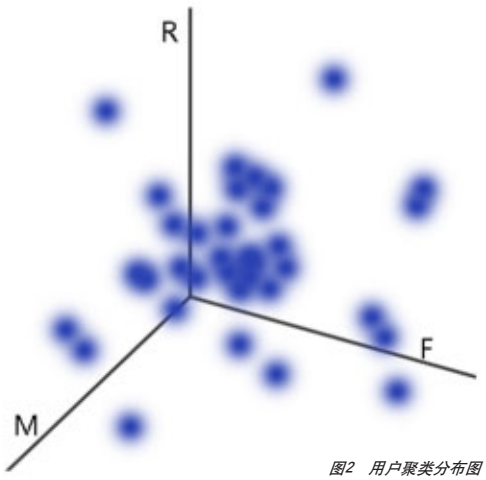


图2 用户聚类分布图

从图1可以看到用户行为密集状况，同样的如果假设每个点的半径变大，我们可以预见到用户分布图中点与点必然产生交集并聚类。半径越大则聚类越少，同一聚类中用户的差异也越大；半径越小则聚类越多，同一聚类中用户的差异也越小，如图2所示。

用户行为聚类的价值在于我们可以清晰地甄别出每种用户类型之间的区别，从而使产品设计更贴

近用户，常见的应用方式包括EDM营销、为不同的用户展示不同的页面和广告等。当然RFM模型只是其中一种方式，还有更多的模型可以进行用户细分，而合理地选择数据模型是技巧之一。

因此，可通过用户数据将用户聚类分组，发现典型用户和构建角色模型等。数据建模不只基于用户行为且也直观地反映了用户行为，所以其准确性和适应性对产品设计有决定性的影响。我们认为行为是产品之魂，数据是产品之魄。

数据是产品的价值表现，但不是价值核心

所有产品经理都希望产品数据非常漂亮，但与此相比，数据的价值更为重要。如果100%坚持数据导向，则不可能产生真正的创新。如果想要真正的创新，则必须坚持用户价值导向。事实上，很多优秀的创新完全抛弃了评估产品的那些数据。假设Twitter完全以PV和IP为导向，那么它一定不可能是今天的样子。

“一切围绕数据”已经成为做产品最容易犯的错误之一，当然我们并不是说数据不重要，而是强调相比用户需求和用户体验而言，数据稍显次要。但与此同时，我们必须认识到数据是用户需求和用户体验的最直观表现，数据虽然是配角但也是最重要的配角。坚持产品以用户价值为中心为产品之道，坚持通过数据分析和挖掘以及其他手段发现深层次的用户价值则为产品之术，以道驭术才能创造成功产品。

利用数据实现最佳用户体验

多年前谈及用户体验就已经产生了诸多“什么样的形式损伤用户体验”之类的争论，几年来这类争论依然不休。事实上，如果项目经验足够丰富且每个项目都认真倾听了用户心声的话，你一定会得出一个结论——好的用户体验形式并无定论。回想很多年前做某银行客户端产品创新项目时，曾有一个没有任何网络经验的老大爷出现在用户研究项目中，而因为其对电脑的不熟悉大大超出了所有在场的用户研究员的预期而此用户研究项目难以进行。也许我们认为此事情有可原，但也许再多几分用心的话实际上这也许会促成一个极有价值的创新。今日重新回归这个案例时，我们不难发现当时存在的核心问题：因为研究时缺乏数据而错误地预设了某类用户成为目标使用人群而导致过于关注“如何让这一类人群最佳使用”而非“如何让存在需求的用户实现价值”。

什么是最佳用户体验将成为永恒的话题。在这个话题上我有几点看法：底层用户体验源自于核心业务能否满足核心需求；基础用户体验源自于核心用户使用是否能够普遍满足；中级用户体验源自于用户使用是否产生了较好的感官体验；高级用户体验源自于是否对每类用户足够细分并且无微不至；更高级的用户体验则来自于超越用户期望的情感体验。

在互联网产品中，真正要做到细致入微，不仅仅需要对客户有200%的了解，还要有能力构建出他们最想要的产品，而其途径则是数据分析。为了让每一类用户都能得到最佳体验，产品的用户体验架构就必须建立在数据的基础上。这样的产品必须细致到细分用户层级，不仅需要非常专业的人机交互功底，而且需要丰富的数据分析经验。当面对数据时，我们首先需要可以通过分析数据特征来构建基本的业务流程，再通过深入挖掘来进行业务细分，这样就可以避免由于概念设计中情景假设失误而导致的业务构建失误问题，而且将数据和业务结合发挥立体效应。在具体进行交互设计时，我们需要把客户特征抽象成为特征数据（比如之前提到的RFM模型抽象也是数据抽象方法的一种），设计多种业务以满足每一类用户的需求（有时用户类型可能多达上百种，如果产品经理能够充分理解数学中的排列组合，那么事情将会变得容易得多），再将这些业务进行合适的规划和关联，形成最优的交互设计地图，如图3所示。

该产品用户需求类型分析，如表1所示。

就图3中的案例而言，我们先通过RFM图形对用户类型进行细分后聚类为ABC三种。在了解ABC类用户的核心需求之后，只需要通过数据挖掘进行细分就可以对应地找出每一类用户，并设计不同

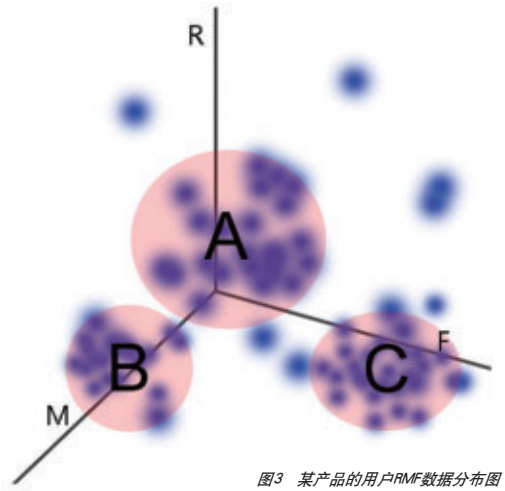


图3　某产品的用户RMF数据分布图

的交互流程以满足不同用户的需求，使得用户在使用过程中如同使用特别定制的产品，用户需求的实现变得更加容易，能够处处体现出对用户的关怀。从另一个角度来说，只有基于数据的用户体验框架才能实现最佳的用户体验。

除此之外，一些常见的数据指标组合也是评价用户体验的重要手段，比如常见的有："UV-UT指标（用户浏览页数和单位浏览时间）"、"IP-PV指标（用户IP和用户浏览页数）"、"转化率-用户忠诚度指标"、"用户活跃度-业务转化率指标"等，这些数据指标直接反映了产品的经营状况，更为直接地反映了产品的用户体验情况，也是用户体验框架中非常重要的一部分。

培养数据意识

伴随着互联网的发展，创新越来越难，而业务要素也越来越复杂，产品经理已经慢慢变成复合型人才了。合格的产品经理不仅仅需要产品设计能力、用户体验功底和数据挖掘和分析能力，还需要有超出一般人的远见卓识。国内大部分产品经理大多是设计或技术背景，数据意识相对不足，这也是大部分企业所面临的问题。如果产品经理缺乏数据意识则会导致整个企业产品以主观思维为主导，产品风险必然成为企业的定时炸弹。

首先，企业必须建立以数据为参考的对话机制。如用户特征、用户行为习惯、用户消费习惯都必须以数据模型的方式进行抽象，产品设计则必须

表1　需求类型分析图表

用户类型（基本数据）	用户关键需求（行为特征数据）			默认交互
	需求A	需求B	需求C	
A	1	3	2	B-C-A
B	2	3	——	B-A
C	1	——	3	C-A

以这些抽象模型为基础。这首先避免了对用户认识的主观化，而且产品设计思路也将更为清晰，甚至于企业的产品运营思路也将会以之为范本。

其次，将产品的用户体验标准数据化作为产品标杆。用户体验虽然是较为主观的内容，但其也可以通过合理量化作为产品标杆。比如用户浏览时间的分布情况、APP流量情况等，这些都可以作为产品设计的方向。但要注意用户体验数据标杆必须适宜且合理，否则反而会产生相反的效果，比如有的产品经理为了提高PV而在不应该分页的地方进行分页，为了加速交互速度而去掉不应该减少的填写字段等。

再次，产品不应该以数据渠道，而应该坚持用户价值驱动。此处要再次重申理Adam Mosseri的产品设计思想：Data informed, not data driven。数据的价值在于发掘用户价值，而不是取代用户价值。产品必须坚持用户为体、数据为用的思想，以道驭术勇于创新，此为产品成功之道。

此外，在过去的大多数案例中，很多企业在执行数据分析方面是消极的，更多的是对数据视而不见，对用户需求的变更视而不见，这不仅是国内互联网企业的通病，也是互联网企业短命的主要原因之一。我们建议数据分析工作的执行程度一定要与产品团队的KPI体系结合起来，形成较好的激励和惩罚机制，这是形成职业化产品团队的一种非常有效的方法之一。



罗旭祥

上海思容信息技术有限公司（思容咨询）总经理，上海益思企业管理咨询公司咨询顾问，广州侨远企业管理咨询公司咨询顾问。著有《精益求精——卓越的互联网产品设计与管理》一书。

产品经理常犯的七大错误

文 / 黄海均



过去一年，产品经理无疑是国内互联网行业最火爆的职位之一。似乎就在一夜之间，挂着这个头衔的人翻了一倍，一些原本做营销策划、运营的人也自称产品经理。更甚的是，不少计算机相关专业的学生都希望毕业之后成为一名产品经理，尽管很多人对于这一职位的工作职责、能力要求知之甚少。除了互联网行业的整体升温、成熟带来的工作细分之外，也有一些是信息不对称造成的。在他们看来，产品经理不需要自己动手写代码、做设计，只需要写写文档、在一边指手划脚即可，而且头衔里有“经理”两字，印在名片上倍儿有面子。

实际上，真正的产品经理并不是一个低门槛的职位。它要求的是能够对一款产品真正负责的人，不仅包括前期的需求调研和分析，中期的产品策划和设计，还有后期的数据分析和产品运营及迭代。从工作内容的广度来看，产品经理对能力的要求比研发工程师和设计师都要高得多。在目前很多互联网公司里，这样一个对公司的产品决策和管理有重要影响的角色，其实际扮演的工作角色更接近于产品设计师，即偏重原型设计、交互设计等，这在一定程度上助长了“人人都是产品经理”的假象。

互联网产品经理不仅工作范围广，职位特点决定了其知识面也要广。有些互联网公司，在招聘产品经理时，除了要求基本的表达和写作能力之外，还重点考察应聘者在设计、心理学、社会学、历史和市场营销等各方面的能力。

“做产品”是一个既感性又理性的过程，纵有很多前辈同行的经验传承和指导，也不乏各种面向产品经理的会议和培训，但落到实处总难免犯一些错误。有些是经验不足所致，而另一些则是产品经理对于“人性”的理解不足。

从我自己的角度来看，做了近三年的互联网产品，的确犯过一些错误，有大有小，道理相通，总结记录让人成长。

1 没想清楚

不知从什么时候起，互联网产品里就开始流行一种“天下武功，唯快不破”的说

法。这个说法误导了很多人，好像只要有一个想法就应该立即动手，边做边改。实则不然，绝大多数失败的产品，问题就出在没有想清楚，比如不确定用户的需求程度，或者做到一半需要架构或需求变更，推翻重来。为了快而使得需求只做了一半，这跟“捡了芝麻，丢了西瓜”没两样。

互联网产品同质化越来越严重，重复造轮子没意义。移动私密社交应用Path 2.0上线之后，国内出现了一窝蜂的抄袭，其实不乏1:1的照搬。我觉得绝大部分Copycat都没有想清楚一件事：在国内做熟人关系的社交产品，如何能摆脱腾讯的阴影？倒不是说只要腾讯在做的事情其他团队都不能做，而是具体到这个点上，其他团队不具备任何优势，漂亮的UI不能解决更多的问题。

产品经理的基本出发点应该是“把事情做对”，而不是“赶出来”。这就要求前期多花一些时间在需求调研上，想清楚再动手，这也防止了后期频繁的需求变更。在《启示录：打造用户喜爱的产品》这本书里，作者Marty Cagan特别强调了充分的需求调研的重要性。

2 想做更多

以微博产品为例，微博解决了用户“我当前在干什么”这个碎片化的表达需求。而从小语文老师就教育我们，一个故事的三要素，应该是“人物、时间和地点”。2010年第3季度，我们想在微博里也同时解决用户“我在哪里”的需求，即发表微博时附带上位置信息，并希望以此构成与竞争对手的差异化。

事后来看，即使这一功能的推出比后来新浪和腾讯微博类似功能早了一年，但这仍然只是一个典型的未经调研的、产品经理自己凭空想象出来的需求。微博既有的使用场景和流程，与签到分享位置无关；且微博里的用户关系半生不熟，向这样一群人分享自己的地理位置，动力也不足。结果显而易见，花了近一个季度做出来的项目，数据表现平平，称之为“失败”并不为过。更大的隐形问题是，产品一旦上线，下线需要更大的决心。这不仅增加了用户使用产品的复杂度，而且带来了难以避免的巨大维护成本。

做加法是容易的，而能够控制欲望“不做什么”更困难，也更重要，产品经理需要有做减法的魄力和决心。当遇到“可加可不加”的功能时，回到原点重新思考，如果不加这个功能会有什么损失？这个事情是否在产品的长远路线上？

3 忽视可用性测试

作为工作的基本门槛，绝大部分产品经理都是产品的“超级用户”。但产品经理并不能以个体代表所有“用户”说话，在很多情况下，产品的目标用户群、用户场景和需求都是多样且复杂的。

因此，尽量早地开始做可用性测试很有必要，用户一定会积极响应。不要因为没有眼动仪就不做用户观察，也不要因为没有用户研究人员就不做用研分析。作为产品经理，少一分矫情，多一分实干，简单绘制原型草图，并直接找几个身边的朋友就能开工。事实上，在网站可用性测试中，一般针对5个人就能发现85%的可用性问题。尤其是在产品定位、产品架构等层面，多与目标用户聊天沟通会有意外的收获。

4 忽略创新的风险和成本

还是以微博为例，评论和转发功能决定了整个社区的信息流通和消费模式，正是由于微博信息的短、平、快等特征，才建立起了微博相对于其他产品形式的核心优势。

而对于微博这种市场竞争激烈的大众产品而言，关于评论、转发功能，竞争对手往往已有了相对成熟的解决方案，这些解决方案经过了用户的验证。越是基础的功能，创新的成本和风险越大，对于产品团队的能力要求也越高。

在这个问题上，网易微博曾有过几次对转发和评论的改版，带来的负面效果也很大，核心用户的流失和骂声都不少。几次折腾之后，现在线上所采用的解决方案依然是照搬了新浪微博的模式，这是最保险和最保守的做法。而几次改版给团队

和用户带来的损失更是难以挽回的。

当然我并不是说产品经理不应该去尝试创新，而是要看清楚这个“创新”在整个产品里处于哪个层级，它对资源、团队能力的要求如何。如果是对产品进行重新定位，服务于另一个用户群或者解决不一样场景下的问题，这样的“创新”往往具备更大的威力。

5 细节决定成败

在中国互联网市场，绝大部分产品都有大量的竞争对手虎视眈眈。在产品大方向正确的前提下，快速上线最简单、最基础的产品版本，并保持迭代小跑比什么都重要。此时不要相信“细节决定成败”，而要相信“战略决定命运”。产品经理不能过度思考和追求完美，要把注意力放在可能会发生的积极事件上，而不要过分关注不太可能会发生的负面事件上。

在微博项目里，早期我花了大量的时间在交互效果上，比如切换页面要最炫的AJAX、新旧微博内容之间的分割线反复雕花、展示微博发布时间的文案反复修改等，而没有将更多的时间花在用户沟通访谈、需求分析与产品讨论上。事后看数据才发现，这些我认为会影响到“用户体验”的细节，只有极少数用户会注意到，这是一种浪费资源和精力的行为。

6 误读数据

数据是对产品运行状况的客观反映，它为找出并优化问题提供了一扇窗口。对于任何一款上线几个月的产品来说，产品经理都可以找到大把数据利用。而在“如何看数据”这件事上，产品经理很容易根据自己的预设来挑选数据，这样导致了所看数据片面和失真的情况，不能真实反应出产品全貌。

2005年，亚马逊CEO Jeff Bezos在致股东信里提到，数据是基于过去的递归呈现，不能为未来的创新带来直接价值。但不可忽视的是，通过详实的数据分析，产品经理可以发现一些新的需求点。

值得注意的是，即使在产品设计前期，产品本身的数据有限甚至为空时，产品经理也可以充分利用搜索引擎、其他数据平台获取到想要的外部数据。大可不必因为“没有数据”而停止往下走。唯一的问题是，想要从那些数据中得到什么？

7 照搬经验

很多产品经理在亲历一两款产品设计之后，都会有一些自己的想法。在下次遇到类似的功能时，会习惯性地照搬经验。但每一款产品的用户需求、使用场景都在时刻变化，即使是自己总结的经验，换一个项目也不能直接使用。依然要经过缜密分析，将事情放在整个产品盘子上，抽身出来客观看之。

实际上，只有极少数人能够将工作中的经历总结提炼成属于自己的方法论，并不断完善饱和。通过与其他人的交流、自己的坚持记录和总结，都可以不断巩固自己的方法论。

以上所写并非全部，回过头来看，有一些即使从头来做也未必能完全做对做好。在特定的时间和空间里，产品经理这个既理性又感性的工作角色，总会面临各种各样的困难，完全杜绝错误并不可能。某种意义上，产品经理唯有积累方法论，规避重复犯错，将事情放在大环境里进行反复斟酌和思考，才能做出尽量接近正确的决定。



黄海均

网络常用ID oxygen，作为产品经理正在为豆瓣添砖加瓦。曾在网易、知乎担任产品经理。

警惕恶性循环

文 / Greg Cohen 译 / 林航



通过对多家公司的长期观察，我发现产品经理的工作中总会存在这样的矛盾：设计了“丰满”的产品特性，却在实施过程中遭遇“骨感”的现实。每当这种情况出现，公司就会陷入“恶性循环”。

开始时，大家都怀着美好的愿景等待产品出世：产品经理高标准、严要求，让开发团队估算出项目的开发时间。如果估算的时间过长，产品经理就会要求团队尽力在项目截止日期前完成，并一再保证该项目计划周详、绝不会出岔子。为了维持和谐的气氛，开发团队咬咬牙勉强保证如期交付。

但在开发过程中，产品经理不断增加新的需求，交付日期却铁打不变。开发团队为了赶工期，需求没被充分理解，最终导致开发出来的产品没能达到产品经理的要求，或是与客户的期望不符。返工在所难免，项目注定要延期。产品经理一着急，怒气全撒在了程序员身上。而程序员想想自己轮轴转地卖力劳动换来的却是指责，于是满腹委屈地抱怨起产品经理，怨他们需求表达不清，想法一变再变。

吃一堑，长一智，经过上一次失败的教训，产品经理在新项目开发前，让开发团队签下“军令状”，要求他们必须按照产品需求文档如期交付产品。可开发团队也不傻，故意延长预估时间，并强调如果需求表述不明，交付时间就无法保证。

这些做法不仅无益于解决问题，反而会加重“病情”。于是，恶性循环开始了。在之后每一次的产品开发过程中，产品经理都会要求研发部门做出更准确的时间预算；开发团队要求产品经理提供更详尽的产品需求文档。情况逐渐恶化，产品经理不仅要设计需求，恨不得连解决方案也一并开发出来；开发团队为了免责，不论是否可行，完全按照需求文档来开发产品。这样下去，只会让客户一再失望，产品无法得到广泛认可。

经过研究，我们发现其实恶性循环源自双方（产品经理和开发团队）都想创造出具有竞争力的产品，并且双方都恪尽职守。殊不知，为了给客户创造价值、给公司谋利益，产品经理和工程师之间的矛盾造成了不必要的“内耗”。

产品经理的职责是发现客户待解决的问题；程序员则要为解决这些问题找准技术途径。双方协同合作，方能做到“其利断金”。

记住，产品经理和程序员是亲密的战友，有着共同的目标。产品经理必须对产品负责，成为恶性循环的终结者，方法很简单：集中精力发现问题，其他的放手交给程序员去解决。产品经理，你们能做到吗？



本文节选自《42 Rules of Product Management》，作者为Brian Lawley、Greg Cohen等人，感谢Happy About出版社对节选内容的中文授权。本书主页为 http://bit.ly/JYmGlo。

走进 MPD，180分钟改变研发团队

走近MPD，充满惊奇的“宝盒”

MPD大会的全称是Make、Professional and Discovery，即改变、专业和探索。翻开教练手册，里面囊括了苹果、微软、IBM、Google、Oracle、惠普、Autodesk、Adobe、SAP等一线研发团队带头人，还有腾讯、华为、百度等一大批来自中国研发军团的佼佼者……

MPD包含了许多企业内部的实战，例如率先推动苹果的Mac Office Excel研发实践、硅谷卓越产品创新体系（PIE），还有腾讯多款互联网产品设计法则；并有机会听到领先产品研发背后的故事，如微软自动化测试体系是如何工作的，Google如何开展Web测试……

第一届MPD大会始于2010年，由msup发起，初衷是立足本土研发团队管理的领袖峰会。首年在北京、上海、珠海大获成功，反响强烈。自2011年调整为区域举办制，按照区域水平特征、研发企业特点设置会议内容，除在北上广一线城市举办外，MPD大会也将走进济南、成都、南京、大连、厦门等地。

180分钟改变研发团队

说起MPD，最值得称赞的当属180分钟的演讲时间限制。对此，品牌拥有者MPD联合创始人刘付强的说法倒显得相当“随意”——“常规会议一般是45分钟左右的Keynote（演讲），内容浅显、时间仓促，但1~2天的公开课对于研发总监或带头人来说太单一、冗长。后来我们研究了软件研发生命周期划分五大角色，并命名会场，把每个题材设置半天（180分钟），每个专场安排角色领域中杰出的4位业界带头人。”

就是这不长也不短的180分钟，创造了无数的“非比寻常”。她就像一个mini电影院，五部精彩绝伦的电影有机结合，20位导演共同演绎研发团队相互协作，以产品经理、团队经理、架构师、开发经理、测试经理作为五个分会场命名，促进角色的共鸣思考、深度探讨、相互交流。每个分会场在课程设计上分别关注了该岗位所必须能力模型：思想、过程、管理、工具（实验室）。MPD选拔优秀的研发带头人来分享自己或所属团队的故事，鼓励改变、专业、探索的软件创新精神，以探寻解决问题的最佳方案。

和谁一起听更重要

当前，软件和信息技术服务业正面临着重大转变，全行业都在加大研发投入，加快企业创新，提升管理能力，研发带头人需要把握趋势、理解需求、重视管理。近两年，MPD听众多以中国软件百强企业为主，其中以中高管理层团队出现较多，交流思想、学习管理，与先进的研发团队共同探讨技术战略与方法对策。

因MPD优秀的议题设置和推广，我们常常可以在会场上看到这些“常客”研发团队：支付宝（中国）有限公司、中兴通讯股份有限公司、艾默生网络能源有限公司、盛大计算机（上海）有限公司、爱立信（中国）通信有限公司、腾讯（深圳）有限公司、中国建设银行软件开发中心、花旗软件技术服务（上海）有限公司、华为技术有限公司、北京百度网讯科技有限公司等，它们已经连续三年组团出现在MPD，课上与知名团队模拟探讨、课下与自己真实团队探讨内部对策。

共享全球智慧“大课堂”

2012年4月21日、22日是个特殊的日子，这是2012年上海MPD举办的时间，也刚好是MPD大会第十届的纪念日。刘付强感慨道：“走过了十届，已发现越来越多人关注研发团队的‘幕后’力量。每一届都是一次革新和飞跃，组织这场小有规模的学习，我们借鉴了大量工作坊教学（workshop）模式，否定了过去填鸭式的会议以及广告味的发布。”

独具特色的五大工作坊

工作坊一：产品创新/用户体验

中国最缺少的就是IT产品经理，而产品经理最缺少的就是新产品的研发与创新能力。该会场覆盖从IT产品预测、构思、设计乃至形成原型这个过程。更进一步，需要继续学习产品的开发实现、产品开发管理流程、上线流程以及运维。部分课程很可能就是学员自己创造一个新的产品的起点。产品经理最难的就是业务

模型设计，这部分的设计高于用户体验式设计。除此之外，还包括市场分析、竞争态势分析、差异化分析、IT互联最技术的走势以及对新产品的影响、未来互联网产品预测、条件约束下的产品设计、产品研发估算、产品需求表达方法、研发管理流程、妥协式设计要点等一系列教学。

工作坊二：团队管理/组织发展

管人是管理的核心。一个好的管理者应该能够激发团队成员的热情，能够帮助团队成员成长。如何激励员工、如何帮助员工成长是本工作坊的主要内容。本工作坊将会以实际的案例和场景展开讨论，内容包括为员工设定提升计划、设定目标、对目标进行跟踪，以及通过各种方式及时准确给员工以反馈。此外，传统IT企业与互联网企业的管理之道，与参与者分享业界卓越的IT信息技术企业标杆企业的战略定位、管理模式和管理流程，其中包括服务型企业的战略定义、项目型企业的全面管理流程以及对软件技术人员的管理方法，包括组织结构的搭建、员工的职业发展规划和绩效考核措施。

工作坊三：架构设计/技术战略

在移动互联网时代，系统开发不仅仅攸关软件设计与开发，还涉及深度软硬整合产品的设计与开发。在当今产业的成功方程式：（（（硬件+ 软件） + 内容）+ 电信）的潮流下，如何将硬件、软件、内容和通信等一群“不会飞”的组件，以一种神奇的创新架构（Architecture）来组织起来，创造有机的整体，像飞机一样“飞”起来。在本工作坊里，将基于新潮的“愿景创新”（Vision-based）架构设计的思维与方法，细腻地将设计师思维与开发工程师思维相互衔接，让学员在分组讨论中渐入佳境，追求设计用户“想摸、想玩、想抱”的亲密IT产品，拓展身为架构师的视野和技能。

工作坊四：开发管理/流程再造

项目管理在传统行业已经非常成熟，可是当它遇到软件行业时却水土不服。在很多大型软件组织机构中，管理是阻碍敏捷软件开发的最大障碍。理解软件开发和敏捷项目管理的本质，掌握项目目标、任务分解、反馈流程制定的基本方法，让团队有一个清晰的管理模型。

在启动一个项目之前，如何以敏捷方式与客户沟通协作、识别项目中的各种风险、确定需求与技术架构、制定项目计划，并就项目的愿景和目标与客户达成一致。演讲将以一个真实的案例讲解如何在客户那里实施Quick Start。

工作坊五：测试管理/质量平台

本工作坊针对在测试领域有一定经验的管理人员工作中常见的八个疑难问题，引用国内外一些最新的观点和做法，给出了一些新的思路和新的做法，以更加深刻地了解测试工作的本质，以及各项指标与工作要求的内涵。这样，他们便可以充满信心、游刃有余地应对每日工作中的各种挑战。除此之外，在分享过程中，讲师还会介绍一些当前世界范围在软件测试领域的最新发展和动态，推进参与者相关的业务水平的进一步发展。

有人预言，用不了十年，北京将成为世界级的研发中心。试想，如果世界五百强的公司都以中国创建研发中心，那么中国作为世界新的科研中心的地位就水到渠成。我们相信这就在不远的未来，而现在，中国软件研发团队正在进行时……





杨钢
金山云CTO

情系网盘

专访金山云CTO杨钢

记者 / 林溪

杨钢，现任金山云CTO。2001年加入金山公司，曾任WPS架构组组长、技术总监，参与过WPS多个版本的开发。2010年，机缘巧合加入《金山快盘》团队，主导团队进行企业内部创业。近期，《程序员》杂志对杨钢进行了专访，邀请他分享自己的创业历程。

机缘巧合加入《金山快盘》团队

《程序员》：请你简单地介绍一下自己。

杨钢：加入《金山快盘》团队之前的10年间，一直在做WPS架构和研发方面的工作。因为机缘巧合，才加入《金山快盘》团队。

《程序员》：是怎样的过程呢？

杨钢：2007年，WPS做互联网转型时，成立了云存储实验室来做一些前沿项目。当时还没有“云”这个概念，把它叫做分布式文件系统。最初将它用在金山毒霸上；后来用在WPS在线存储上，即在WPS上加网络存储功能。大家发现，用户对“随时随地可用”这种理念非常认同，方才逐渐意识到，这项技术用在WPS里有点大材小用，然后决定做《金山快盘》这种产品形态。由于我是后来加入这个项目的，所以我猜当时是借鉴了Dropbox的思路。

2010年初正式立项做《金山快盘》。当时云存储方面已有现成的技术，只需做客户端，仅过4个月，就开始做产品公测。之后，大家很快意识到，这应该是个会高速发展的业务，放在WPS这种迭代周期相对较长的体系里，会阻碍它的发展速度。因此，决定将《金山快盘》从WPS中独立出来。

公测时发现，服务器端不稳定。而我当时在服务器管理部门，所以被调到了这个团队。我一下被《金山快盘》吸引住了，开始兴奋且专注地改善服务器端的稳定性，几乎把整个服务器端架构全部调整了一遍。

此时，WPS的CEO发了一封邮件问我，假设让我来为《金山快盘》的发展下注，我愿意投多少钱？看完邮件，我直接回的价格几乎是我当时拥有的所有财产。从那之后，公司就安排我来主导《金山快盘》这个项目了。公司最初并没有想以内部创业的形式来做《金山快盘》，因为后来看到除我之外其他团队成员也都愿意为这个项目投钱，才决定将《金山快盘》作为一个创业项目。2011年植树节前夕，我们从珠海搬到北京，正式组建创业团队。

这就是《金山快盘》的由来，以及我加入这个团队的过程。

一起拼的感觉，终身难忘

《程序员》：组建这个创业团队时，有没有遇到什么困难？

杨钢：因为是企业内创业，所以组建团队时，

需要所有团队成员投钱。我比较干脆，想都没想就同意了。而其他的团队成员，还是经历了一番思想斗争的，但最终大家还是认可了团队的方向。在保障基本生活的前提下，很多人几乎把这些年的积蓄全都投在了这个项目上。

来北京的这个决定是北京公司要求的。我之前也想过离开珠海，因为感觉珠海太养人了，比较适合养老，所以他们一提，我就答应了，算是一拍即合。团队的其他人在这件事上有些犹豫，我给大家做了一番思想工作。不过还是损失了将近一半的成员。

《程序员》：在这一年多时间里，你觉得最难忘的事是什么？

杨钢：最难忘的是刚来北京租用民房办公的那段日子。团队搬到北京之后，我们做了一个很明智的决定——没有进金山办公楼，而是在小区里租民房办公。租的是一间复式的半地下室，环境比较艰苦，不过幸亏当时走了这一步。

我在金山公司工作的前10年基本上每天上下班很准时，有很多的业余时间去做自己喜欢的事，比如学习第二外语或者做其他有意思的事。我想，团队里很多成员都和我差不多。然而这种有点艰苦的环境一下子激发了大家的“战斗”热情。工作环境变了之后，大家的工作节奏和工作方式也随之发生了很大的变化，团队的凝聚力一下子增强很多。

大家每天都在全心地研究产品、做开发，几乎没有时间去管外面是白天还是黑夜。而且我们的家人都在珠海，要很长时间才能回一次家，确实牺牲挺大。这段种创业的氛围和与兄弟们一起奋斗的感觉是这辈子最难忘的，而产品技术层面的问题反倒是其次的。

租用民房办公的时间将近一年。后来因为民房的电不够用，而我们在工作时需要用计算机，而且由于人员方面的扩充，房间变得很挤，连走路都困难了，整个团队才搬到金山办公楼。刚搬进来时，还有一个笑话。我们第一天就被投诉了，说太吵。现在大家都很安静，怕吵到隔壁的部门。

说句实话，创业有时就是头脑发热，不能按逻辑的观点来思考。现在想想，我们当时的投入还是很可观的，无论是成本，还有所付出的努力，基本上等于前几年白干了，全部从零开始。但我认为经过这种创业氛围，团队的价值得到了提升，是个不小的突破。

创业心得分享

《程序员》：能分享一下创业心得吗？

杨钢：我认为团队决策不够快。之前，我们基本上是按照大公司的方式来决策，有时还要论证，决策太慢。现在看来，创业有时就是在拼，认准一个方向，然后拼命去试，成功与否不重要，重要的是尽快获得结果。

另外一个问题是，我们做的东西太多。实际上，去年已经被取消了很多项目。这并不是研发能力的问题，也不是技术上的问题，而是一个团队不可能同时做好那么多事。这应该是所有创业团队都会遇到的问题——什么都想做，以为自己什么都能做。

还有个问题就是在引进人才方面做得不够好。今年会在这方面投入较多精力，吸收一些有经验且能够认同双方的人。去年，大学生比较多，中层管理人员相对欠缺，梯队管理不完善。

这几点问题应该是所有创业团队都会面临的问题，相信会慢慢得到解决。

去年有一件让我觉得有些苦恼的事——团队一直没有CEO。而我把自己定位为CTO和代理CEO，在没有CEO时凝聚这个团队。我认为自己做得还不错，只有一名团队成员离开，但几个月后又回来了。在创业团队中，我们算是稳定性比较高的团队。

但我在杂事上花费了较多时间，比如跑融资等，在技术上投入的时间相对较少。而我认为自己能为团队贡献的最大价值在于技术。今年团队已有CEO，我会把管理方面的事全部交出去，专心做技术。

经过一年多的积累和努力，相信《金山快盘》会在今年快步发展。

时间的旅行者

计划与估算导论

文 / 殷亮

有机体越高级，它的行动计划便越复杂，它的每一部分在其地点和时序方面也就越全面地依赖所有其他部分，以体现它对整体的用途。

——Michael Polanyi

人：时间旅行者

在地球上曾生活过数十亿的物种，甚至出现过像暴龙、巨齿鲨这样不可一世的绝对霸主。但无论是哪一物种，都不曾像今天的人类这样绝对地统治地球。科学家将之归结于人类卓越的智慧——从某种意义上说，从人类第一次举起棍棒开始，曾显赫一时的生物力量就已经变得微不足道。而这种智力进化的一个显著标志，就是生物体的自我意识，在此我们可以套用生物演化学家Theodosius Dobzhansky的名言：“自知是人类的根本特征之一”，由此确立了自我意识在人类史上的重要地位。

自我意识的出现意味着个体意识到了一个恒常自我的存在，它构成了人类区分“我”与世界的这种能力的基础，也促成了我们在时间上进行回忆与展望的可能。心理学家Rollo May曾如此论述：“它（自我意识）赋予了人类留住时间的能力，这是一种超脱于当前、想象昨天或后天的自己的能力。因此，人类能够从过去中进行学习，并为将来做出计划。”由此，我们转到了本文将要讨论的主题：计划。如前所述，计划之所以可能，是因为我们能够脱离自我巢臼、站到一边，来审视我们的历史；是因为我们能够身不动却心已远，去思考我们的未来；同时，也是因为我们能够像他人看待我们那样来看待自己，可以对他人进行移情。这份天赋让我们终成自然的征服者，飞天揽月、统治地球；也使我们能做时间的旅行者，活在未来、行动在现在。

计划：难臻完美

作为时间旅行者，我们能想象将要到来的一次会晤，甚至能在想象中思考自己将如何行动、他人是如何回应的等。尽管这不过只是一种看似寻常的简单计划，但其本质上与指挥一场会战的军事家的运筹帷幄并无不同，都是行为人在时间上向前展望的结果。行为人借此超越了时间和空间的限制，对将要发生的事情提前进行思考、预测与规划，从而在心理上缓解了事件发生时临时进行决策的压力与焦虑感，同时避免了因仓促决策而带来的经济上的各种损失。

然而，由于计划在很大程度上仍是基于我们对未来事件的主观预期，因此随着时间跨度的拉长，我们远期的计划相比起近期计划必然会有更大程度的失据。因为预期的时间越长，可能对结果造

成影响的外在事件就越多，而我们所能预见的往往不过是所有这些事件中的冰山一角。再加上制订远期计划所需的信息很多时候都只有在前面的工作完成后才能得到，这也使得我们很难在一开始就为项目制订一个全面的计划、并将所有的可能结果都包含进去。因此，当目标还很遥远时，我们只能依靠有限的信息与预测向这个目标的方向前行。

这是这个变动不居的世界自身的复杂性使然。我们对于世间万物不完备的知识及所掌握的有限因果规律，都还远远不足以让我们对遥远的将来作出准确预测。近一个世纪以来，时间见证了人们对计划认识的变迁——从最初强调计划的完备性与理性，到今天对计划难臻完美的坦承；从对甘特图、负荷图、PERT网络分析这类计划工具的争相追捧，到现在人们对白板、随手贴这类低技术含量工具返璞归真式的青睐——这一过程无形之中又与科学发展的一般规律悄然吻合，即总是从一个极端或另一个极端开始，然后再经历不断地修正，逐步向中间发展。

从工程计划到敏捷计划

今天，人们之所以渐渐接受了计划的不完备性，是因为已经意识到：与其说不完美的是计划，倒不如说是不完美的人造就了不完美的计划。于是，在工程领域，几十年前人们便开始采用一种叫叠浪式计划的方法。以欧洲北海油田的项目为例，这种计划方法会先对计划进行分层：第1层是概要计划，这个层次的计划显示的仅仅是一些主要里程碑事件，用来向业主或承包商的领导层汇报；第2层计划则是对第1层的扩展，包含了项目结构，从中可以了解到项目各组成部分之间的关系；而第3层计划则用于控制详细的施工进度，通常也是最低层次的整体计划；到第4层时，计划是用工作卡片来描述的，该层计划通常是针对小的工作群体或工作要素的计划，细化到天。当然，这4层计划并不是一开始就能被详细地定义出来，而是一个类似波浪式的过程：在项目的开始阶段，人们一般只能制定第1层的概要性计划，对整个项目作出一个大纲性的安排，同时制定出早期阶段的第2层、第3层计划；然后，有更多的信息（如生产率、结构设计等）在早期阶段中产生，便可以对整个项目制定一个第2层次的计划，同时，对项目中期阶段制定一个第3层次的计划；最后，当更多的信息在项目中期产生后，项目最后阶段的第3层计划便可以制订了。这一过程就好比一个滚动向前的波浪，从左到右移动，因此被形象地称之为叠浪式计划。

如果将这一方法应用到软件开发中来，那么第1层无疑就是产品或项目层次的各个里程碑，像RUP中的初始阶段、细化阶段、构建阶段与移交阶段。在项目最开始时，我们一般都只能对这些里程碑进行大致的估算，并对早期阶段的计划进行细化；而后续阶段的详细计划，如架构工作如何开展、做哪些技术预研等，这些安排只有在早期阶段完成后才会细化出来；类似的，也只有在细化阶段完成后，后面的迭代构建计划才得以确立。熟悉Scrum敏捷方法的人或许会很快发现，Scrum中的Product Backlog与Sprint Backlog也正是体现了这种结构上的层次和时间上的渐进。由此可见，工程项目也好，软件项目也好，尽管两者之间存在着一定的差异，但这些差异并没有我们想象的那么大。至少在对项目的计划和控制方面，两者体现了明显的殊途同归的迹象。另外，对项目灵活性的重视，以及尽可能晚地制订详细计划的原则，既非敏捷方法原创，也非软件项目独有。这一结论无疑会让那些固守着专业疆土、决不承认自己的领域与其他领域有相似之处的人难以接受，但管理的本质就是如此——管理理论在抽象层次上与具体的行业无关。

估算未来

因为计划本质上的不完备性，无形之中也给管理者带来了一系列的难题，尤其是影响了计划的微观层面。例如，工期估算问题，至今都仍没有与其重要性相匹配的成熟理论。如果计划是完备的，那么这些微观问题将不复存在，人们只需努力地追求完美的计划即可。但在现实世界里，这无疑是奢谈，只要这个世界还存在不确定性，那么能让人一劳永逸的完美计划就永远只是管理者

可望而不可及的海市蜃楼。与此同时，由于理论领域对这一层次问题的关注是如此零碎而又缺乏深度，我们也因而在很大程度上面临着知识的匮乏。但我们仍然需要去思考这些问题、去怀疑已有的答案，否则我们就只会被淹没在那些对问题毫无洞见的陈腔滥调之中。

2006年，Steve McConnell出版的图书《Software Estimation》，被认为是业界翘首以盼之作，并获得了次年的Jolt大奖。当然，考虑到软件估算的重要性，这无疑是实至名归的。毕竟如果像统计数据所报告的那样——在所有项目中，差不多只有1/4按时交付，近1/4的项目被取消，而大约一半的项目在交付时超时、超预算或两者都超过，那么对于软件项目管理人员的估算能力，我们并没有理由乐观。

在《Software Estimation》中，McConnell对一些重要的估算公式进行了介绍，其中包括PERT的预期计算公式（1.1）：

$$\text{预期情况} = \left[\text{最好情况} + \left(4 \times \text{最可能情况}\right) + \text{最差情况}\right] / 6$$

这个公式并不难懂，它的意图是，在估算过程中同时考虑最好情况、最差情况与最可能情况，这样就避免了只考虑单一情况所可能造成的偏差。这一方法对于那些忽略多种可能性的计划而言，至少是一个不小的进步。当然，这并不是唯一的计算方法，McConnell继续介绍道：“有些估算专家建议改变基本的PERT公式来考虑估算值偏低的情况”，下面是改变后的公式（1.2）：

$$\text{预期情况} = \left[\text{最好情况} + \left(3 \times \text{最可能情况}\right) + \left(2 \times \text{最差情况}\right)\right] / 6$$

调整后的公式加大了最差情况的估算权重，但是否就一定比之前的公式1.1有更准确的结果呢？作者并没有给出明确的答案。最近几年，也经常有人询问我究竟哪一种方法更好，希望能有一个明确的结论。但实际上，不管是公式1.1，还是公式1.2，本质上是没区别的。通过数学转换，我们可以将上述两个公式变为以下形式（1.3）：

$$\text{预期情况} = \frac{1}{6} \times \text{最好情况} + \frac{x}{6} \times \text{最可能情况} + \frac{5-x}{6} \times \text{最差情况}$$

甚至还可以进一步一般化为（1.4）：

$$\text{预期情况} = \sum_{i=1}^{n} \text{情况 i 的发生概率} \times \text{情况 i}$$

现在我们可以看到，这无非就是一种对概率的加权平均：让i种事件与其对应的概率相乘然后求和。但对于软件开发任务而言，实际上可能的情况远远不止最好、最可能和最差这三种，以统计学的观点看，这更像是连续概率分布而不是离散概率分布。因为理论上可能的情况是无限的，我们不能列出所有可能的取值；而且由于存在无限多的情况，所以任何一种情况的概率事实上是无限接近于零的，从而我们只能计算一个范围取值的概率。也就是说，在前面的估算公式里，最好情况、最差情况和最有可能情况是不能指向某一个点的，而应该是指向一个范围，因为无论在哪一个点上，发生概率都不会达到1/6。

举例来说，假如一个任务我们估算它的最好情况、最有可能情况、最差情况分别是5天、8天、12天，那么对于公式1.1，它的意义是什么呢？以最好情况为例，公式1.1在最好情形5天上乘以1/6，这并不是说任务在5天这个点上完成的概率是1/6，而是说在计划者在任务工期的概率分布上划分了一个称之为“最好情况”的范围，其面积为整体概率面积的1/6，在这个范围里，5天是其均值。

一言以蔽之，这里的最好情形也好、最差情形也好，并不是日常语义上的同义词，它们的取值也并不是我们现实意义中所描述的那个时间点，而是一个指定的概率范围内概率均值所对应的点——理解这一点是尤为重要的，否则我们便无法从数理上对公式1.1和1.2作出解释。因此，对于每一个估算者而言，重要的不是选择公式1.1还是1.2，而是在你和你的团队的共同假设里，你们所定义的最好情况、最可能情况及最差情况在概率形状上是如何分布的，它们的面积是否与对应的乘数一致？在这一方面，团队必须有非常清晰的认识并达成一致。只要能做到这一点，在对最坏情况加权时，我们便有充足的理由选择小于1的任何数，而不是仅限于1/6或2/6，如图1所示。

我们在项目计划上的失败，有很大一部分都可以追溯到估算上的拙劣表现，而这些拙劣表现，又大多与我们对统计学的似懂非懂不无关系。当为项目设置缓冲区时，我们总是倾向于认为各个任务的延期或提前在大数定理（Law of large

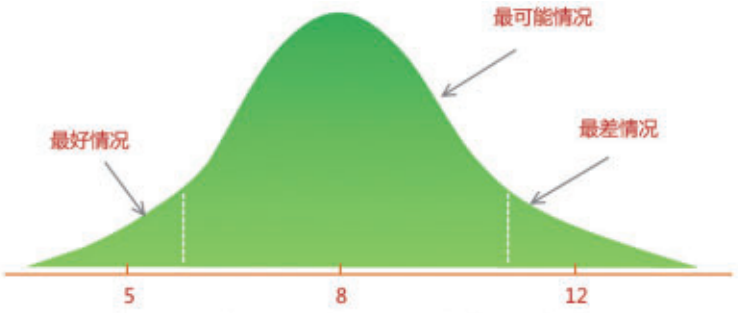
numbers）的作用下会相互抵消，从而不需要太长的缓冲时间。但殊不知，样本只有在容量足够大的情况下其均值才会接近于整体均值：当向天空抛1000次硬币，我们可以合理期望正面与反面的频数各占一半，但并不能期望在只抛2次的情况下也能有同样的结果。统计学权威Calyampudi Radhakrishna Rao曾如此感叹：“对统计学的一知半解常常会造成不必要的上当受骗，而对统计学的一概排斥也往往造成不必要的愚昧无知。”今天，我们已经普遍懂得了数据的重要性，也被各种各样的数据所围绕，数据就如同思维的时光机，让我们的思维可以在弹指间便涌至遥远的将来。但要达到一个正确的将来，我们还需要用更多的时间去分析、过滤数据和解释数据，而不只是接触数据这么简单。

超越自然选择

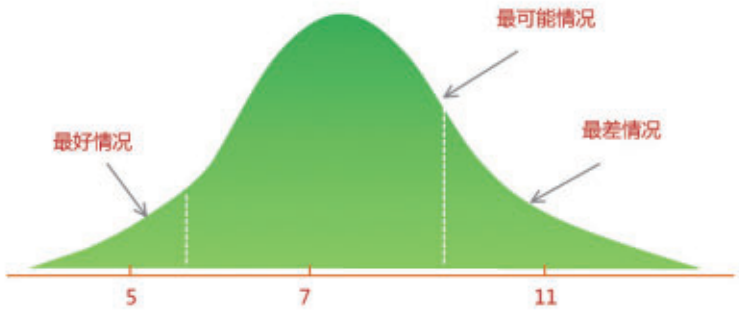
自我意识的存在，使我们为未知的将来制订计划成为可能，也使我们在这种茫茫黑暗中仍能发出内在的微光以照亮前面的方向。正如前面所说的，这是一种天赋，借助它，人类得以超越时间与空间的限制；但与此同时，这也是一种诅咒，因为它，我们才开始感知焦虑与孤独、承受生与死的对抗。

在进化史上，自然选择的过程是机械化的、短视的，“进化机器”会接受任何随机产生的突变，只要能赋予生物体更强的繁殖能力就行，而不管长期结果如何。它不具备有等待、迂回或为了未来而投资的能力，但人类的行为却能借助计划作出更好的选择——我们有能力设想实际上尚未发生的变异，可以推迟获取报酬并为了将来而投资，能彻底改变环境而不只是去适应环境。

今天，宣称计划无用或轻视计划的声音一直都不绝于耳，当中不乏借着对有机体、开放式系统的一知半解对计划进行抨击与否定的偏激者。我认为这很大程度上在于他们还没有真正理解计划的本质和意义，因为他们还不知道：他们所排斥的、非难的，却正是他们每一步行动都在遵从的、每一天生活都不可或缺的东西。在经济学里，作为抽象演绎部分的理论经济学，只是经济学帝国广阔疆土中的一小块封地，是经济学的众多分支之一，同时它与真实世界的差异也一直饱受非议。但在某种意义上，它却是“第一分支”，是经济学其他部分的根基与先决条件。对于计划而言，这一点也同样成立。计划只是我们行动中的一小步，却是第一步。



公式 1.1 预期情况=[最好情况+(4×最可能情况)+最差情况]/6



公式 1.2 预期情况=[最好情况+(4×最可能情况)+(2×最差情况)]/6

图1 对比公式1.1和1.2



殷亮

目前在深圳工作。研究领域以产业竞争与组织理论为主，6年软件项目管理与组织顾问经验。

责任编辑：杨爽（yangshuang@csdn.net）

敏捷中的控制，控制中的敏捷

文 / 许正华

本文在简要介绍施耐德企业文化模型中的控制文化的基础上，探讨了两种不同类型的控制——垂直控制与水平控制。随后分析了水平控制对敏捷的影响，并为其实践提供了若干指导。

有人说中国文化不适合实施敏捷，因为敏捷强调自组织，而中国人历来缺乏自我意识，习惯于被动的接受控制。我部分赞同上述观点，但也觉得这样说未免将这个问题过于简单化了，因为敏捷对于任何组织都是一个巨大的挑战。如何理解控制文化以及它在敏捷中所扮演的作用是一个至关重要的问题，值得我们来深入探讨一番。

施耐德文化模型中的控制

施耐德（Schneider）在《The Reengineering Alternative: A plan for making your current culture work》一书中描述了协作、控制、培养、竞争等四种典型的企业文化，如图1所示。

图1中右上象限描绘了施耐德模型中控制文化的主要特点：充满了政策与过程；致力于创建一个清晰而有序的组织结构；关注的是公司/系统及当前的状态；强调权利、秩序、可预测性和稳定性；坚信组织的成功源于保持对每一件事情的控制。对大部分公司而言，这个场景实在普通得不值一提。

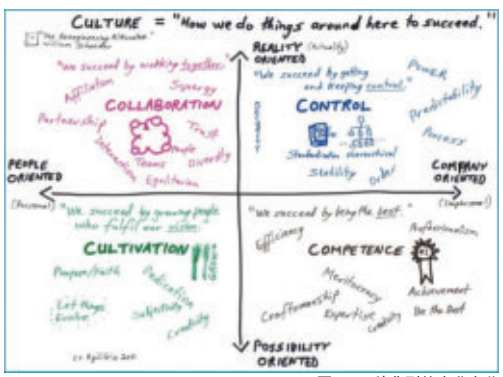


图1　四种典型的企业文化

“打破科层制后的迷失”

我所在的企业是一家传统的软件开发公司，控制文化无处不在，并占据统治地位。而随着敏捷的深入人心，团队间的合作确实比以前变得更加紧密了，但同时也出现了另外一种现象，我把它叫做“打破科层制后的迷失”。让我们先来看看两个实例吧。

实例1：工程师小强找工程师小李讨论一个设计问题。小李认真地思考后提出了自己的意见。小强没有积极回应，还是沉浸在自己的思路里。最后，小李愤愤地说“你自己看着办吧，反正我的观点已经说了！”

实例2：作为Scrum Master，我每个月末都会将下个月计划完成的功能列表提前发给产品经理进行确认。通常情况下，我不会收到任何实质性的反馈。这并不出乎我的意料——很有可能因为忙于其他更重要的事情，产品经理并没有认真阅读我的邮件。我的内心有一个声音在回荡“已经发了

邮件，还要我做什么！”。最终我欣然将“没有反馈”等同于“通过审查”。

我们似乎都已经尽了自己的职责，说了该说的话、做了该做的事，但问题并没有解决。当传统的软件研发组织结构（科层制）被类似Scrum的敏捷团队取代后，该如何建立一套新的秩序呢？

两种控制文化

层级式的控制文化

施耐德文化模型中的控制文化是层级式的控制文化，控制是通过清晰的组织结构自上而下单向传导的，它的载体是指令。军队是层级式控制文化最浓重的组织之一。“服从”无一例外地成为了各个国家的军队里最重要的核心价值之一，无论它穿上什么样的外衣。

对任何一个软件组织而言，单纯依靠层级式控制是无法有效运行的。最主要的原因非“变化”莫属——需求不断变化、设计不断调整、实现不断演进。计划似乎总追赶不上变化，每个人的工作也相应地变得有弹性和不可预期。变化是普遍存在的——即使是军队也会因变化不得不“将在外，军令有所不受”。这预示着在一个有效运行的组织中还应该依赖于其他一些东西。

自主式的控制文化

你是否也遇到过下面的场景：两个人在面对面地讨论一个问题。讨论初期，气氛很融洽；随着讨论的深入，开始遇到一些分歧，语速越来越快；双方都在试图说服对方接受自己的观点，嗓门越来越大；到最后，只剩下两张眉头紧锁、肌肉僵硬的面庞。我就曾经是这样的参与者，后来我知道控制对方语速的最好方式就是放慢自己的语速，控制对方嗓门的最好方式是降低自己的声音，控制对方情绪的最好方式是让自己始终面带笑容——虽然有些难度。

在这个例子中我们看到了另外一种有别于层级式控制的行为模式——自主式控制文化。我们可以通过一系列“是”与“不是”来理解自主式控制的内涵。

- 自主式控制是自发的，而不是盲从的，更不是被迫的（至少不是被他人所迫）。
- 自主式控制基于契约（共同目标），而不是等级制度。
- 自主式控制的传播途径是行动，而不是指令。
- 自主式控制讲的是阳谋，而不是阴谋。

对沟通的控制源于自我对成为有效沟通者的渴望。沟通的目标可以是多种多样的，但绝不会是让别人接受自己的观点，因为没有人不能保证自己的观点永远是对的。所有参与者都是平等的，因此最好的控制是通过我的积极行动实现的。在沟通中，合理运用肢体语言是一种简单而有效的策略。

二元控制文化

为了方便记忆，我分别为层级式的控制与自主式的控制取了别名，前者叫作“垂直控制”，后者叫作“水平控制”。前面我们提到过即使是纪律严明的军队也无法仅仅依靠垂直控制，那么水平控制是否可以完全掌握一个软件组织呢？开源社区应该算是这个领域内最松散的组织形态，即便如此，它也有自己的核心领导团队。核心团队居于整个社区的中心，是非制度化的、动态的社群领导者。

任何组织都离不开“垂直控制”与“水平控制”相融合的二元控制文化。足球是一项非常讲究控制的竞技运动。它里面包含着层次式的控制结构，守门员、后卫、中锋、前锋各司其职。理想情况下，皮球经过守门员、后卫、中锋、前锋层层渗透，最终进入对方的球门。但我们都知道真正的比赛比上述描述复杂许多，里面穿插着变化无穷的水平控制。场上每个人都尝试着以自己的方式解读和控制比赛。

关于控制的若干误解

长久以来，我们的生活中存在着一些对“控制”的误解，它们深刻影响着人们的思想和行动。你认同下面这些观点吗？控制是个人意志的灌输；控制的最终目标是为了实现个人价值的最大化；控制是权力者的阴谋与游戏。

存在这些误解与垂直控制文化的盛行不无关系。

表1 常用的控制策略

策略名称	描述	实例
灯塔策略	防止迷失的重要手段就是在众人面前点上一盏明灯	如何控制好会议进程？让会议目标得到与会者的认可是会议组织者与参与者的共同责任
启发策略	永远不要低估榜样的力量	在公司的广告语策划案中，我们团队的提案是最多的，并且其中一位员工的提案被最终采纳。与其他团队相比我们并没有什么过人之处，有的只是相互间的影响与感染，因此我们应特别感谢最初的几个提案者
激励策略	认可与赞许可以让你事半功倍。生活中并不缺少美，缺少的是发现美的眼睛	N/A
大炮策略	有时控制力不够，是因为力量被太多的目标所分散。这时可以考虑用炮弹取代散弹	在确认需求时，我改变了以往的批处理策略：每次只和产品经理确认有限的几个即将开始开发的功能

在垂直控制中，控制者与决策者往往是同一个人，缺乏共同目标的形成过程。并不是说垂直控制中没有共同目标，只是决策者手中的权利使得个人意志与共同目标的差别对于大多数人变得模糊而不重要了。而水平控制中，控制者不必是决策者，任何人都可以成为控制者。

在这样的误解下生活久了，人们变得羞于控制、懒于控制和不善于控制自己的工作与生活。我们可以容忍自己的职业发展充满疑惑与抱怨，却不愿作出自己的抉择。

自主控制让敏捷团队更有效

敏捷团队需要自主控制

敏捷更适合于以人为本的团队文化，重要的问题都是依靠集体智慧来解决的，因此人的状态对敏捷团队的成功非常重要。自主控制可以让敏捷团队更有效，有以下两方面的原因。

■ **自主控制是积极心里的源泉**。心理学研究表明，恐惧容易让人们在面对困难时选择逃避，这是很多心理疾病产生的根源。在软件开发过程中，很多负面情绪的产生都是因为没有及时实施积极的自主控制，最终导致局面失控。通过讨论我们能发现很多原以为无法控制的改变，其实是可以通过积极的自主控制加以避免的。

■ **自主控制是协作的基础**。敏捷强调协作众人皆知，但人们往往会忽略协作中的控制因素。简单的协作并不能提高团队的效率与凝聚力，如果处理不好反而会降低团队效率与和谐。因为协作不会是一帆风顺的，能否顺利化解困难考验着团队成员的坚韧。倡导自主控制文化有助于培养队员坚韧的品格。

有效的自主控制

可将自主控制能力由低到高分为以下三个等级。

■ **控制环境能力**。这里所说的环境指办公物理环境、同事关系、工作内容等诸多方面。

■ **实现目标能力**。利用已有的资源，在当前能力范围内建立并实现共同目标的能力。

■ **领导变革能力**。开发潜在资源，提升自己的影响力和领导力，帮助团队达成共同愿景的能力。

能否持续地进行积极的自我控制是有效的自主控制的基础。一个人如果不能控制好自己的小环境，也就谈不到控制大环境了；一个人如果不能明确自己的目标，也就谈不到建立共同目标了；一个人如果没有勇气与能力塑造自我，也就谈不到领导变革了。自主控制是讲策略的，表1中是实战中我经常用到的控制策略，供大家参考。

总结

当科层制被打破后，控制仍是必要的；控制的具体方式，由指示变成了推动协作的阳谋；控制力构成了领导力的一个组成部分，进而成为影响敏捷团队与文化的“人的因素”的一个部分。而我的讨论，便仅止于此。我并不试图去判断本文所指出的“解”的正确或合理与否，我只指出这一个解的全貌，以便于读者从整体（而非具体的手法）上去认识它。



许正华

目前就职于CA Technologies中国技术中心，从事数据备份和高可用企业软件方面的开发和研究，并专注于软件工程、知识工程、团队文化等领域的研究与实践。

责任编辑：杨爽（yangshuang@csdn.net）

寻找机遇 创造未来

CSDN 热门职位全新推荐

■详细信息请参见CSDN网站

GAMELOFT

历经十年，Gameloft 已发展成为一家国际领先的出版商和视频游戏开发商。从手机到 iPod 再到便携设备、电视游戏机，Gameloft 的游戏遍及全球。

现诚聘以下职位：

- C++ 软件工程师
- Java 软件工程师
- PHP 软件工程师
- 原创游戏策划人员
- 游戏制作人（Producer）
- 2D 游戏美术设计
- 3D 游戏美术设计

简历投递邮箱：recruitmentBJ@gameloft.com
网址：www.gameloft.com

新华网股份有限公司

新华网是由新华社主办的大型网络文化企业，现因事业快速发展诚聘英才。

现诚聘如下职位：

- 高级 Java 开发工程师
- 高级 Web 前端开发工程师
- iPhone 开发工程师
- Android 开发工程师
- 研发工程师（分布式搜索方向）
- 产品设计师

地址：北京大兴国家新媒体产业基地 星光影视园 10 号
简历投递邮箱：dev@xinhuanet.com **网址：**www.xinhuanet.com

新东方教育科技（集团）有限公司

新东方教育科技集团，18 年办学经验，至 2011 年底培训学员达到 1500 万人，业务涵盖早教、学前、小学、中学、大学、在职、出国考试 & 留学申请、英语能力提高、小语种学习、国内 / 海外夏冬令营、图书、在线教育等，旨在为学员提供一站式终身学习服务。

现诚聘如下职位：

- .NET 软件工程师
- .NET 高级工程师
- PHP 软件工程师
- PHP 高级工程师
- 前端开发工程师
- 网站实施工程师
- 软件测试工程师
- 数据分析工程师

简历投递邮箱：zhaopin@xdf.cn，请以“CSDN+ 姓名 + 职位”为标题

ThoughtWorks

如果你**热爱代码如生命**，喜欢有事没事就记录和分享自己的思考与实践，希望远离办公室政治，追求公平合理，追求成为一名优秀的程序猿、攻城狮。那无容置疑 ThoughtWorks 绝对适合你！
你将和一群 Geek **在编码中一较高下**！
你将和优秀**的咨询师**一起，享受在帮助客户解决棘手问题之后的成就感！
你将在不同国家、不同项目、不同技术中**快速成长**！
工作地点：**北京、西安、成都、上海**

现诚聘以下职位：

- Java/C#/Ruby 开发工程师
- 质量保证工程
- 前端开发工程师

简历投递邮箱：job@thoughtworks.com

北京用友华表软件技术有限公司

用友华表自 2011 年起，定位于用友集团 BI 平台、工具及各领域 BI 分析产品的开发和营销。华表公司是目前国内知名 BI 软件厂商，并将在未来两年内打造国际领先的商业智能平台。

现诚聘如下职位：

- 数据仓库技术架构师
- 多维分析技术架构师
- 数据挖掘技术架构师
- Java 高级开发工程师
- VC 工程师
- Android 开发工程师
- iPhone 开发工程师
- Java 集成工程师
- 应用开发工程师
- ETL 架构工程师

地址：中国北京海淀区北清路 68 号 用友软件园 A 座 3 层 1 区
简历投递邮箱：hbhr@ufida.com.cn **网站：**www.cellsoft.cc
电话：8610-62432270

长沙聚游动漫设计有限责任公司

公司的产品定位为游戏机开发，经营理念以市场为中心，充分发挥员工的创造性和积极性，为员工提供一个不断发展的平台，与公司共同成长。

现诚聘以下职位：

- 客户端主程序员：
 5 年游戏开发，2 年客户端主程序经验，主导开发过完整的游戏项目；熟悉一种主流的游戏引擎；使用 C++。
- 客户端游戏程序员：
 具备一年以上单机游戏或网络客户端开发经验；熟悉 D3D 或 OpenGL 的使用，使用 C++。

地址：湖南省长沙市人民路 29 号颐美园 6 栋 1202
简历投递邮箱：csjydm@sohu.com **电话：**0731-82088789

寻找机遇 创造未来 CSDN 热门职位全新推荐

■详细信息请参见CSDN网站

深圳市明源软件股份有限公司

深圳市明源软件公司，始终专注于房地产行业信息化领域，是目前国内领先的房地产应用软件及解决方案供应商。

现诚聘如下职位：

- iPhone 开发工程师
- .NET 开发工程师
- 软件测试工程师
- 软件设计 / 需求分析工程师
- 高级开发工程师
- 项目经理

地址：深圳市南山区高新科技园南区高新南一道中科大厦
简历投递邮箱：hr@mysoft.com.cn
网址：http://www.mysoft.com.cn

广州赫基信息科技有限公司

赫基信息科技是赫基（香港）集团旗下分公司， 主要负责集团线上业务及打造电子商务平台。赫基集团为业界精英创造充满前景的广阔天地，提供个性化的福利、透明的培训晋升机制。

现诚聘如下职位：

- 产品经理
- 研发工程师
- 一线支持工程师
- 软件架构师
- 系统工程师
- 运营工程师
- 测试工程师
- UE 高级设计师

地址：广州市天河区林和西路 161 号中泰国际广场 B 座 36 楼
简历投递邮箱：RecruitEC@trendy-global.com
网址：www.trendy-global.com

广州多益网络科技有限公司

多益网络，创立于 2006 年 7 月，专注于网络游戏研发与运营，拥有国内位居行业前列的研发运营团队及一流的产品研发能力，是广州领先的专业网游研发及运营公司。

现诚聘如下职位：

- 游戏服务端程序员
- C++ 软件工程师
- 游戏客户端程序员
- 网站技术经理
- C++ 高级程序员
- asp.net 高级程序员

地址：广州市天河区翰景路 1 号金星大厦 3-7 楼
简历投递网址：hr.henhaoji.com
网址：www.henhaoji.com

埃菲柯德信息技术（北京）有限公司

中外合资企业，总部位于芬兰的创新之都赫尔辛基。2010 年成为全部研发基于敏捷开发并通过 ISO9001 认证的软件公司。

现诚聘如下职位：

- 销售经理 (5 年 IT 销售经验)
- Windows Phone 7 软件工程师
- 客户经理 (3 年 IT 销售经验)
- Android 软件工程师
- Qt 开发工程师
- iPhone 软件工程师
- S60 研发工程师

地址：北京朝阳区望京地区
简历投递邮箱：hr.china@eficode.com（标明所申请职位 +CSDN）
网址：www.eficode.com

上海交通大学E-learning Lab

上海交通大学 E-learning 实验室是上海交通大学优秀的独立研究中心。开展各种领先性技术研究，旨在推进 E-learning 教学领域内的理论、技术及产业化发展。

现诚聘如下职位：

- 高级软件研发工程师
- 视频编解码技术研发工程师
- 高级 Web 应用研发工程师

地址：上海市华山路 1954 号交通大学浩然高科技大厦 6 楼
简历投递邮箱：viena@sjtu.edu.cn
网址：dlc.sjtu.edu.cn

浪潮集团通用软件有限公司

浪潮集团是中国知名的软硬件整体解决方案供应商，是我国云计算、物联网、三网融合等方面的龙头企业。浪潮通软作为浪潮集团核心产业之一，是著名的 ERP 软件及服务供应商，在咨询服务、软件开发及解决方案提供和实施服务等方面具有强大优势。

现诚聘如下职位：

- BPM 高级工程师
- e-HR 系统分析师
- Web 应用开发高级工程师
- Silverlight 应用开发工程师
- 数据访问引擎高级工程师
- .NET 软件工程师
- ESB 高级工程师

地址：山东济南高新区浪潮路 1036 号
简历投递邮箱：gshr@inspur.com 网址：www.inspur.com

北京中油瑞飞信息技术有限责任公司

中国石油集团旗下专业 IT 技术服务供应商。2006 年成立至今已形成软件服务、ERP 咨询、财务金融、油田应用、海外业务、系统集成、信息安全、数据中心八大业务板块，员工 2200 余人，拥有信息系统集成二级，CMMI 三级和 ISO20000 等多项国际资质。

招聘职位：

- 系统工程师（系统运维、监控实施）
- 云计算系统工程师（软件、虚拟化、网络、存储、服务器、综合、安全、核心技术、综合）
- 软件工程师（JAVA、.NET、BI 产品研发）

简历投递邮箱：richfit_hr@cnpc.com.cn 请以“应聘职位 + 姓名”命名
电话：010-59981054、59981057 网址：www.richfit.com

聚胜万合信息技术（上海）有限公司

聚胜万合创建于 2009 年初，并在当年获得美国光速创投基金的首轮融资。公司聚合了业界优秀的互联网技术、广告营销、互动创意专家，是专业从事精准营销及数字营销的专业广告技术和服务机构。

现诚聘以下职位（工作地点：上海）：

- 云计算工程师
- 前端开发工程师 /PHP 软件工程师
- 资深系统运维工程师
- DBA
- 前端开发工程师（CSS+JavaScript）

地址：上海市闸北区恒丰路 568 号 恒汇国际大厦 10 楼
简历投递邮箱：hr@mediav.com 网址：www.mediav.com

广州瀚信通信科技股份有限公司

现诚聘如下职位：

- C/C++ 中级开发工程师
- C/C++ 高级开发工程师
- C/C++ 系统架构师
- C# .NET 初级开发工程师
- C# .NET 中级开发工程师
- C# .NET 高级开发工程师
- C# .NET 系统架构师
- 研发项目经理（C/C++ 或 .NET、Delphi 等均可）

简历投递邮箱：yfhr@hantele.com，请以“职位 + 姓名 +CSDN”作为邮件标题

北京法国电信研发中心

Software engineer in the web service system development

Position requirements:

- Master degree in telecommunication, Compute Science OR equivalent
- Experience in J2EE, SOA, Spring, Hibernate, Struts, XML, REST, HTML5 development
- Experience in cloud computing development, such as SaaS, PaaS, etc.
- Strong programming skills, at least 3+ years experience in Java software development
- Familiar with the software development methodology
- Familiar with networking technology
- Good English communication skill
- Good team working spirit and communication skills, and hard working as well

简历投递邮箱：hr@orange-ftgroup.com.cn

网易有道

作为网易自主研发的全新中文搜索引擎，有道搜索致力于为互联网用户提供更快更好的中文搜索服务。依托网易强大的产品服务平台和丰富的资源优势，有道搜索吸纳了众多优秀的创新人才，并正在高速发展中。

现诚聘如下职位：

- 前端开发工程师
- 测试开发工程师
- Web 开发工程师
- 研发工程师
- 有道笔记 / 有道词典 UI 设计师

北京市海淀区中关村东路 1 号清华科技园 3 号楼（创业大厦）2 层
简历投递邮箱：wangjian@rd.netease.com 网址：www.youdao.co

上海征途信息技术有限公司

上海巨人网络科技有限公司成立于 2004 年 11 月 18 日，是以网络游戏为发展起点，集研发、运营、销售为一体的互动娱乐企业。

现诚聘如下职位：

- 资深客户端软件工程师
- 资深服务器端软件工程师
- 资深 3D 引擎软件工程师
- 资深 Java 游戏开发工程师
- 资深 Flash 游戏开发工程师
- Web 前台开发工程师
- 资深 Web 开发工程师（PHP）
- Java 软件开发工程师

地址：上海市徐汇区宜山路 700 号 3 号楼
简历投递邮箱：hr@ztgame.com 网址：hr.ztgame.com

马拉松前100米的风景

《智能360》创始人李传丰专访

记者 / 陈粲然

2011年10月12日，苹果开放了iOS 5操作系统的更新，在其诸多的新特性中，Siri是最令人瞩目的。这项由人工智能技术支持的声控型虚拟个人助手服务，通过对对话、自然语言理解、视觉、演说、机器学习、制定计划、理性思考等功能与互联网服务和本地功能的整合，为移动互联网打开了新的入口。令人遗憾的是，Siri目前并不支持中文，同时Google也未在Android系统上推出类似服务，面对新接入方式带来的无限遐想空间，中国的开发者们怎甘落后？2012年伊始，各种“中文语音助手”应用如雨后春笋般涌现。本文的主角，聚熵信息技术（上海）有限公司是国内第一家发布中文语音助手（2012年1月15日正式上线）——《智能360》的开发团队，然而与其他竞争者相比，他们并没有太多的先发优势，并且与大部分年轻的创业团队相似，他们也面临着缺乏资金、渠道等各方面资源的痛苦。这样一个在移动互联网中的较新领域里探索的典型创业团队，其成长思路值得每一位开发者深思。

团队的“化学反应”至关重要

《智能360》的几位创始人中，何永和李传丰曾一起考北大研究生，后来两人分别去了北京和上海的两所大学读书。2009年，还在读研一的两个年轻人就开始创业了，他们的首个创业项目是基于SNS模式的考研社区，经验的缺乏导致了这次创业的失败。此后，何永选择了休学参加工作，先后在淘米、第九城市、乐哇等公司作为研发工程师历练了一段时间。

2011年年初，何永、李传丰以及现在的COO李育华、CTO廖益玄利用业余时间开始第二次创业。当时大家都是兼职创业，李育华在第九城市移动互联网事业部担任运营商渠道运营数据分析师，廖益玄在搜狐负责Android视频客户端播放器模块的设计与开发。

有效的“化学反应”直接决定着团队能够一起走多远。一支创业团队很难做到阵容豪华，但成员之间一定要相互信任、坦诚，且知识结构及能力互补。用李传丰的话说：“我们一起打过仗、创过业、失败过，彼此之间相互了解、配合默契，每个人在自己适合的位置上，整合力远大于个人能力。”

在进入移动互联网的初期，并没有一个非常理想的切入点可供这支小团队选择。他们最初的产品是《语音360》，它的定位是一个语音功能聚合平台，主要功能有语音备忘、通话录音、电话留言、语音播报以及来电提醒等。这是一个相对小众的市场，想象空间也有限，但他们认为，如果等到条件都成熟了，再去做某个市场，小团队是没有任何机会的，创业者应该勇于尝试、快速试错、

适时调整方向、不断完善，以打造成熟的产品。

2003年创立，直到2009年才依靠《Angry Birds》起飞的开发商Rovio；6年烧掉1700万美元，推出35款游戏却无一成功的《Draw Something》开发商OMGPOP，都是经长期修行集大成者的代表。而许多中国的开发者还是想得太多、做得太少。

由于Siri的备受亲睐，以及中文智能语音App上的空白，团队决定将开发重心转向中文智能语音助手领域，从《语音360》中将语音助手的功能提取出来，在中文智能语音助手这一新兴市场上全力一搏。

专心致志练内功

对创业公司来说，“智能语音助手”是一个门槛较高的创业方向。语义识别、人工智能这些核心技术，都为这支年轻的团队带来极大的挑战。

李传丰坦言，现在的技术水准离“人工智能”的美好愿景还相差甚远。人类为什么聪明？因为我们不仅掌握了足够多的知识，还拥有很强的思考能力。所以对于《智能360》来说，首先要做的就是其知识库的构建（自学习系统的构建以及完善），使它短时间获取海量结构化高质量数据，只有在这个前提上，它才有可能“聪明”；除此之外，还需要不断强化自然语言处理引擎，构建一整套强大的算法，准确识别用户意图。这二者缺一不可。

“这将是一个漫长的积累过程。好比一场马拉松赛跑，目前我们只是在前100米暂时领先”，李传丰很形象地表述了他对先发优势的理解。

正是因为知识积累需要大量的数据，所以聚熵团队在《智能360》中加入了“学习”功能，希望能够依托用户的力量建立海量数据库，帮助它一步步向“智能”的方向迈进。

“我们本着实事求是的态度，不会为了给用户答案而出现答案，不知为不知，不能很好地解决问题时就不要不懂装懂，我们宁可让《智能360》在现阶段显得笨一些，这样更利于它的成长”，李传丰的解释十分坦诚。踏踏实实地练内功，是他们当前的心态。



这支年轻团队需要专心练内功

需要依靠巨头扩大市场

2012年，国内移动互联网投资有所减缓，移动互联网第一波以工具型应用的占位争夺高潮已过，而且大多最终会变成巨头的游戏。那么巨头的进入是否会对聚熵这样的小团队造成冲击呢？在交谈中李传丰并未表示这方面的担忧，相反地，他认为智能语音助手类应用有着广阔的市场空间，只有大家合力才能把蛋糕做大。

李传丰说：“有竞争是好事，能够激励团队持续创新、快速迭代。只要我们能够贴近用户，把用户的需求快速反馈，就能拥有我们的市场。由于市场机会巨大，且各公司的定位也不相同，所以不必过分强调所谓正面冲突，恰恰相反，冲突会带来更多的思路和学习榜样。”

不过，作为一支年轻的创业团队，他们也有自己的烦恼。目前他们还未花钱做过推广，而一些竞争对手已通过各种渠道开始宣传甚至预装。虽说好产品会说话，但在这个同质化严重的时代，移动应用的分发是不可或缺的力量。

聚熵团队每天都在上海闵行区一个安静的居民区内踏踏实实地做着产品，与他们相似，在上海、杭州、广州、成都、北京、武汉……还有千千万万个开发团队正经历着满怀憧憬的日日夜夜，他们推动着移动互联网的高速发展，愿他们在创造更多好产品的同时抓住自己的梦。

声效表达在游戏沉浸方面的影响力

文 / 郑金条

作者在2012年《程序员》5月刊中着力分析了游戏用户留存的两个隐性层面：情感牵制力和社区牵制力，本文将会从声效表达上，以共鸣层、陈述层、形态变化层、技术层和方案解决层五个角度进行匹配解析。

共鸣层

Danger Inc音效设计师Peter Drescher在分析游戏多层面对玩家的吸引力时认为大量的游戏都做得相当精致，并且总是抱着无限的期待希望玩家能够持续地在游戏中投入时间甚至金钱。但这往往又无心地忽略了一个最严峻的层面：玩家并不总是喜欢那些无限制反复播放的音乐，抛开音效是否真的和游戏本身匹配得亲密无缝不说，单是无所顾忌的重复播放就足够将赏心悦目败坏为最恶俗的艺术以致玩家要么怨弃音乐要么远离游戏。

spoomusic网站音乐总监Ariel Gross认为声效的这种场景因素能够将玩家的临境感通过听觉的迷幻陶醉带入到更为逼真和投入的层面。就如音效设计师Marc Lamber所称的环境音效强调是游戏音效中最别致贴心的一点，例如玩家突然从地下室入口走出来第一刻见到阳光时那种雀跃或身处于一个让人惶恐之处时那种毛骨悚然的心慌，如果类似这样的特殊感受被无限循环的曲目以不着调的方式一带而过，那种剥离感可以深刻到让玩家主动去逃离游戏。

最直接呈现的是K. Hew、M. R. Gibbs和G. Wadley在墨尔本大学所作的一项关于玩家对听觉腻烦的忍受度分析，事实上当这种效果达到实质性地让玩家烦恼时，他们的第一行为就是抛弃原本是促进游戏交互和沉浸的听觉交流方式。这种描述类似于Peter Drescher所归纳的，做游戏的声效需要具有相应的游戏临场感，展示具体的场景需求，将声效愉悦转归为玩家的听觉享受而不是苦恼于各种简单生硬的或者无节制重复的播放效果。

或者可以从专业执行者的角度来看待声效对共鸣沉浸的影响。Splash Damage音效总监Chris Sweetman认为声效甚至能够可以很好地提升玩家的游戏意愿，以致有些看起来一般的游戏在音乐的陪衬下突然显得有十足的把玩性，就如Freelance音效设计师Jerry Ibbotson在探讨《MotorStorm Arctic Edge》所称道的关于汽车换档时以引擎的声效来提示玩家游戏正在带来不断加速的冲刺愉悦。

形态变化层

有两个趋势的演变其实是不利于音乐元素在游戏中的强渗透表达的，一个是规模化的团队正在为小作坊式的框架所取代；另一个是Unity、Unreal或者Torque等便捷工具正在被更广泛地运用。

Free Range Games音乐总监Brad Meyer很清晰地表达了这个层面，在SNS和移动游戏释放了游戏

独特创建的时代新纪元的同时，小预算几乎让很多研发团队阻隔开了研发和声效配置本身，更多地表现为音乐部分大多为外包服务，甚至是远程服务，而有些音乐的外包商本身并不参与游戏的全程运作，只是在必需的情况下额外介入，既可能不关心细节，也同样可能无法理解某些场景的当刻需求，还可能因为外包资金需求不匹配而产生的各种敷衍行为，以致游戏出现各种非针对性的声效灌输，或者不够协调或者对游戏的美感没有丝毫的提升效果。

就像我们早先所强调的声效或者音乐是游戏无处不在的细节一样，开发者会为游戏达到视觉画面的美轮美奂投入无限的精力，或者在前端和后端的程序运营或者服务器负荷方面做到没有瑕疵，但在音乐层面却在表现力和游戏的匹配层方面吝啬投入。要知道开发者最大的奢望是玩家在体验游戏后能够长时间地留存下来并时刻牵挂着回访或者无限骄傲地向自己的好友推荐和分享自己的游戏心得和欢乐，缺乏针对性和表达力以及无限重复的音效有可能在听觉方面第一时间败坏了玩家的游戏兴致。

随着游戏工业的进程所验证和展示的，声效并不是可有可无的附属成分，声效所营造的是深入心扉的体验氛围，渗透开发者十足情感的音效将在捕捉用户灵魂和沉浸度方面更具魔力。或许工具和技术手段可以带来不错的声效体验，但只有跟上游戏进程的专业音效师才能更好地理解游戏。

陈述层

就现在来讲，着实有两个层面可以印证音乐的表达力在游戏中有不可取代的效能。

其一是游戏音乐从业者的薪酬范畴，按照Game Developer调查数据（2500份的数据取样），2011年美国游戏声效师的薪酬均值为83182美元（相比2010年的68088美元提升近1.5万美元），高于游戏行业平均薪酬的81192美元。如果从岗位分工看，2011年，游戏声效师的薪酬低于运营、程序员和游戏制作人但高于美术动画、游戏设计师和QA从业者；相较于2010年，游戏声效师的薪酬待遇有了最大幅度的提升，2010年，游戏声效师的薪酬为68088美元，仅仅高于QA从业者的49009美元，远低于运营、程序员、美术动画和制作人。



《Civilization IV》声效获得格莱美奖

一年的变化，让游戏声效师的价值报酬超越了美术动画和游戏设计师（相比较2011和2010，后两者的薪酬提升也在3000～4000美元之间，但落后于游戏声效师将近1.5万美元的幅度）。

其二是游戏音乐在产业中的独立效能，典型的包括Christopher Tin凭借Sid Meier《Civilization IV》声效获得格莱美Best Instrumental Arrangement Accompanying Vocalists；而Zynga旗下的经典力作《Castleville》的原创音乐插曲已经提交iTunes Music Store独立出售，定价为0.99美元；早先《Farmville》巅峰期时，Steve Kirk制作的《Farmville Song》只有很短暂的30秒钟，却可能成为当时每天播放量最大的音乐曲子之一（数千万次）；拓展远一些还可看到各种不同的顶尖音乐人都试图和游戏产生营销关联，诸如Bon Jovi、Grateful Dead、Lady Gaga、Dr. Dre。

目前游戏业的顶尖音乐人包括《Plants vs. Zombies》的Laura Shigihara、《Super Mario Bros》的Koji Kondo、《Dead Space》的Jason Graves、《Final Fantasy》的Nobuo Uematsu、《Call of Duty》的Sean Murray、《Red Dead Redemption》的Woody Jackson以及《Prince of Persia》的Penka Kouneva等。

技术层

《游戏声音结构》的作者Sander Huiberts从IEZA（Interface、Effect、Zone、Affect）四个层面来解析游戏的音乐结构和最终的声效问题。在他的观点中，游戏的声效分为两个环节，一个环节是游戏的自然来源，主要附着为游戏的背景乐或者与实际场景氛围相挂钩的背景音效，诸如雨天的声音、城市嘈杂的声音或者丛林兽鸣以及鸟鸣等声音源；另一个环节则和玩家的游戏操作行为休戚相关，属于游戏的玩家表达反馈，诸如游戏角色行走发出的声音、枪械战役中的打击声、汽车引擎的发动声或者游戏中的物件碰撞等可选择操作的声音源。前者渗透在游戏每一个细微的进程，后者则由游戏事件或者玩家的直接操作行为引发产生。

这就涉及到游戏声效/音乐的逻辑呈现问题，例如如何在技术层面不影响游戏流畅度的情况下保障这些声效能够从更系统性的角度得到更好的表达（假定声效部分已经完美实现，只需要通过技术层面进行合理嵌合，并能够在玩家需求的位置得到良好的表达展示）。按照Game Developer Magazine特约编辑Damian Kastbauer的话讲，如果技术层面处理配合度不好，再好的音效也可能变成一般或略显低品质的听觉感受，这是对音频团队和整体游戏效果处理默契配合度的考验。

Lionhead音效总监Russel Shaw在谈到类似的问题时认为，首先要建造一个合理的系统将环境音效层更加真实地加载在游戏地图中，例如当玩家开始在丛林中跑动时，游戏就适时地将音效处理调整到丛林环境之下，而当游戏环境切换到流水旁时，又必须准确地将声效定位在流水声的氛围内，这个就必然需要从整个技术层的实现可能进行考虑。

因为任何一个环节的差异对玩家来讲都可能产生临境感的差异，例如玩家处在流水旁所展示的却是车辆轰鸣的引擎声或者喇叭声就显得相对失真。Raymond Usher曾从声音的层面来考量过玩家在游戏进程中生理方面的变化问题，引述的例证包括有恐怖探索氛围的《Amnesia：The Dark Descent》和有各种嘈杂环节的Flatout系列游戏。接受试验的玩家需要分别接受有声环境和无声环境的情绪测试，对参与者的心率和呼吸频率的分析显示（包括皮肤温度），那些玩带有音频的游戏的人拥有更高层次的激发状态，由此呈现了游戏中音频促进玩家融入的能力。

方案层

Clefbits创始人Marcelo Martins曾专门就游戏中曲目循环播放的历史问题进行了回顾，认为在预算有限的情况下，大部分的游戏都会选择曲目循环播放的策略，尽管众所周知的原因，循环曲目在一遍遍的循环往复之后终究可能导致玩家的集体唾弃。他给出的方案是消弭音乐中的起点和终点对音效做无缝衔接，降低音乐节段的识别度，在循环播放的流程中从形式上获得玩家的差异认同造成音乐比实际长度长相当多的错觉，当然在无限的循环之后玩家的警觉性仍然会强烈地意识到这个频段有限的音乐只是在做重复播放的机械动作，而非感觉上的不同曲目的跳跃。

DSK Supinfocom分析师Philippe Vachey的方案是适当整合进静音（音效间歇、音高或者音频）来环节循环曲目中时刻的高相似度（事实上因为游戏存储空间和为音效预留的预算有限，重复曲目似乎已经成为了某种必然需要直面的问题），并进行细节调整从间歇层面、频率层面微调，从差异化的角度继续为娱乐玩家创造空间。Cipher Prime Studios音效设计师Dain Saint认为从情感控制和诱发的角度来说，音效首先都不能成为玩家觉得厌弃的对象，与之相反则需要表达出加分渲染的效果，诸如在有悲伤氛围的地方，一旦加上贴切的哀婉的音调，则整个气氛就立刻能够感染到玩家本身了。



郑金条

游戏邦负责人，游戏邦主要关注和解析国内外社交游戏和手机游戏领域，并定期做深度行业阐述。

责任编辑：陈博（chenbo@csdn.net）

狂奔的移动端用户体验设计（下）

文 / 石爽

移动终端面临着复杂的环境，作者以实例为基础，详细从可控感、速度感、好感化三大方向，阐述了如何抓住用户体验设计的重心。

在复杂的环境下开展用户体验设计，需要明确几个重心，并围绕重心开展具体的设计工作。在过去的两年中，我们结合实际项目，边实践边总结，找到了一些可供大家借鉴的设计思路。

可控感

所谓可控感，即：通过设计手段，让用户享受掌控的感受。操作简单、容易理解、意料之中、误操作少……简单就意味着控制【注：引自Giles Colborne的《Simple and Usable Web, Mobile, and Interaction Design》】。

■ **精简内容层级**。在移动端进行产品架构的设计时，一级导航分类的定义要清晰明确，让用户第一时间知道App的核心功能；一级导航不超过5个，过多反而会造成选择和记忆困惑；一级导航避免只有图标没有文字注释，减少用户猜测和回忆；产品层级设定最好不超过3层，过多会让用户迷失自己当前的位置。

■ **使用用户的语言做设计**。《可用性工程》一书中提到，“作为以用户为中心的设计的一部分，用户界面中的词汇应当使用用户的语言而不是面向系统的术语”。简单来说就是站在用户的视角来描述事物。例如使用常用词语，但并不意味着局限在常用词语。如果用户对系统的专业名词也很熟悉，我们也要使用系统文案，统一对公共概念的认识。同时，用户的语言不仅限于文案，还有图标符号。

避免使用独有的、专有的名词文案、符号，因为这会让用户摸不着头脑，找不到本源。有时保持



图1　从入口到详情，两个层级完成对产品的架构



图2　从人机界面的四个基础维度对设计语言进行统一体验设计

与系统一致，也能降低用户的学习成本。文案、图标的表达表意设计工作，是我们每个项目必须要做的事情，它们是用户与机器互动的关键媒介。

移动端的百度搜索在色彩、符号、文案、交互四个基础维度，向百度PC大搜索靠齐，形成了标准色彩体系、专属符号库、文案设计指导原则、搜索交互模式文档，这些在新时期共同形成了用户对百度搜索的体验感受——统一、易用、便捷。

■ **关注主线任务，二八法则分配**。主线任务是该应用的核心功能，也是大多数用户常用的功能。专注核心任务，是提升产品价值的主要途径。主线任务的清晰，决定了用户心目中绝大部分的体验感受。

围绕主线任务开展用户体验，可以避免一味累加次要功能和高级功能，造成入口或者操作繁复，干扰主线任务的识别和连贯，最终导致应用在用户心目中产生模糊的概念。

百度搜索手机客户端的主线任务是输入搜索，支线流程是语音搜索、拍照搜索。因此在框的设计区域，我们更多地强调输入框的视觉比例。同时交互体验升级重点也都集中在输入搜索。例如：增加搜索历史、搜索建议、搜索建议到输入框上。视觉上，分类图标弱化，突出输入框。下文会有对输入搜索核心任务优化的说明。

百度搜索手机客户端的浏览框架主要解决用户浏览结果的需求，因此没有将浏览做得大而全，没有增加高级的浏览功能，选择前进、后退、刷新等常用的浏览功能，满足大多数情况的浏览需求即可。

■ **巧用点触区域**。我们知道，触屏手机的点触操作区域是有一个基本标准——44x44dp。这在Android 4.0手机上是48dp。点触区域大于这个范围，用户更容易点击，误操作少。

在我们的实际项目中，视觉控件可以适当小于点触区域，而点触区域要达到44x44dp标准。这样给用户带来了更易操控的体验，同时界面的元素比例也更加轻盈。

iPhone输入法键盘会随着用户点击按键，预判可能点击的下一个按钮，并适当增加点击的区域。这在一定程度上，让输入准确率大大提升。

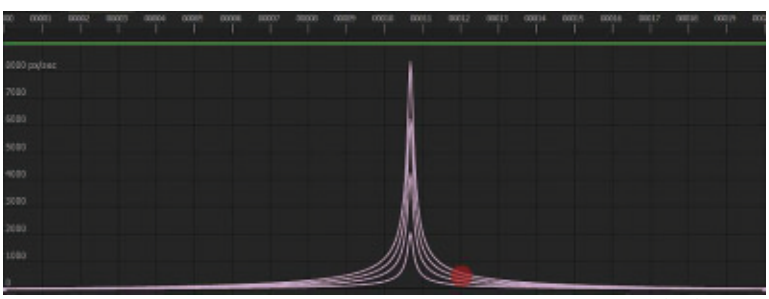


图3　菜单按钮弹出的动势

速度感

速度感即为效率感受，在界面切换、服务响应、操作反馈、信息加载、误操作回馈、等待动画等多个模块中体现。速度感由时间和感受两部分组成。例如一个控件的动画，时间要适当控制不可太长，效率类应用最怕动画还要看一两秒；而控件的动势会在同样的时间内，让用户会感觉到慢或者快。速度感是工具类应用的核心竞争力。

我们在实际的产品设计中，专门针对效率感受进行深入设计。

■ **进度条**。数据加载的等待感受，可以被进度条的动画取代。一定程度上缓解了用户对等待加载的焦虑情绪。

在百度搜索App项目中，结果页面的加载用到了进度条这种常规控件。当大量用户反馈界面加载慢时，很多情况是受到网络环境的限制导致的。我们使用心理学实验，测试进度条的动画路径，来指导视觉的设计。实验结果表明“先慢后快”是用户体验感受快的方案，也就是说，后期加速对降低主观时间感受有正面的作用。

■ **避免冗余动画，学会利用加速度**。用户喜欢动画效果。华丽的切换、展开、收起……但同时我们不能忘记保证效率。好的动画效果用在无形之中，辅助UI框架的呈现，让信息在方寸间流淌。在设计动画时，我们针对动画的加速度进行微调，让用户感受到轻微的吸附力。菜单按钮弹出的动势，在时间和速率的共同作用下，表现出了积极、正面的动画效果，如图3所示。

参考网页中的响应时间规律，一般而言，在0.1秒

内响应，用户是可以接受的。1秒是用户保持不间断的思维流的限定时间，如果超过了0.1秒而少于1秒内响应，用户会产生疑惑。因此需要显示将要完成的时间（通常选择进度条或百分比来表示），不然会大大降低用户的期待值。在移动端，由于用户对网络有心理预期，所以用户对响应的时间会相对长些，苹果认为10～20秒为最长的限制，越快的响应才会产生越积极的影响。

■ **自动判断还是直接呈现**。在条形码识别搜索项目中，用户使用手机扫描条形码时，不再需要拍照这一动作，只需将手机摄像头对准条形码，机器开始实时捕捉，一旦确认条目清晰立即呈现条码结果，将拍照和识别交付给了机器。

自动判断也需要慎重使用。在语音搜索模式下，由于大部分硬件还不具备消除环境音，机器经常误判断用户一直在说话，所以我们增加了“完毕”按钮，让用户来决定是否结束录音环节。

■ **基于效率体验的设计案例**。百度搜索交互体验设计的首要环节是输入。从输入的方式来说，目前有按键输入、语音输入、拍照输入。

目前大多数用户还在使用按键输入，内置的输入法在输入中文多个字的关键词时效率不高，输入法联想词无法满足用户输入精准联想的需求。我们在联想词、输入键盘等多方面的整体解决方案的打包融合一获得更好的输入体验，如图4所示。

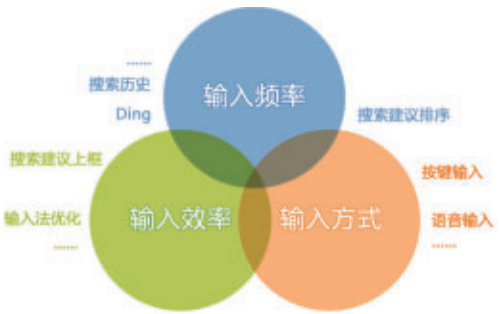


图4　百度搜索手机客户端的输入法打包

好感化

设计是工业设计上比较流行的设计理念。苹果公司是个好榜样，好的产品设计包含了一半交互一半心理学。用情感驱动设计，并创造一个良性循环的可持续的情感设计机制，会为品牌带来潜移默化的效果。

让用户愿意体验、想要体验、等待体验。这是产品发展到成熟阶段的现象。在不同阶段，好感化设计的重点不同。百度搜索App先期的好感化设计方案集中在效率、强大、自然、轻盈、探索这五个正面情感。

效率——如前面所讲的，包含了核心任务的交互效率提升、效率可视化设计。

强大——搜索客户端具备强大的后台技术，前端也要显示出来，并让用户因此而感受到。

自然——我们利用应用大面积的留白背景，进行了壁纸的主题设计。将星辰、晨曦、海边、植物等自然画面引入其中，与人崇尚自然的内心互动。同时这个背景也很好地与Ding进行信息展现互动。

轻盈——将操控感更强的UI引入进来，剔透而轻薄的操控感受，让整个App具有灵动的生命。

探索——优秀的语音搜索、图像搜索让用户尝鲜体验，让用户开始尝试用更多的方式发现搜索引擎的魅力。我们在语音搜索方面的动效设计、图形化设计，也都具备一定的新颖和科技感。

总地来说，面向移动端的用户体验设计，要针对复杂设备、复杂环境提供解决方案，抓住可控感、速度感、好感化三个设计重心，与用户一起成长。随着大量IT巨头进入移动互联网，更多的跨屏幕、跨平台的服务将会大规模出现，更多的设计难题开始涌现。在这个大环境下，搭好“用户—设备—服务”之间的桥梁，是用户体验设计的终极目标。



石爽

百度移动·云事业部用户体验部设计经理，从事互联网体验设计工作7年。

责任编辑：陈博（chenbo@csdn.net）

Cocos2D家族的前世今生

记者 / 陈粲然

Cocos2D，这款用于开发2D游戏的开发框架以及由其衍生的Cocos2D-X跨平台开发框架和众多分支，正在帮助越来越多的开发者实现高效稳定的游戏开发。《程序员》近日专访了Cocos2D-iPhone作者Ricardo Quesada，请他分享Cocos2D家族的发展历程与未来方向。

1973年，Ricardo Quesada出生于阿根廷。

在Ricardo 12岁时，他母亲为他购买了第一台电脑，一台Commodore 64。

刚开始，与其他孩子一样，Ricardo将自己的精力放在电子游戏上，但在几个月后，他购买了编程方面的书籍，开始学习用BASIC语言编写一些简单的游戏。两年后，Ricardo开始学习汇编语言。

在Commodore 64上，Ricardo制作了很多游戏并贡献了一个游戏编辑器，而这台Commodore 64也陪伴了他长达6年之久。此后，Ricardo选择了iAPX 286作为他的新伙伴。在他学习了C语言后的几个月，Ricardo在DOS系统下编写了一些游戏Demo和一个游戏编辑器。

当然，他并不满足于此，随后Ricardo又开始涉足Linux系统，并将编辑器移植到Linux系统上，将汇编语言改成了C语言。Ricardo将Linux作为主要的操作系统用了很多年，并不断地为开源社区贡献代码。此后他与一些志同道合的朋友一起学习了C++以及Python语言，并参加了一些开发大赛。在大学期间，Ricardo选择了钟爱的计算机科学专业，并不断涉猎计算机安全、算法、游戏架构、OpenGL等大量书籍。

Cocos2D的诞生

2005年，Ricardo和朋友萌生了用Python语言“一星期编写一个游戏”的想法。在2005～2007年间，他们设计了许多种这样的游戏。值得注意的是，每次在设计一个新的游戏时，其游戏引擎都是重新开发的。

2008年2月，他们在阿根廷Córdoba市附近的Los Cocos，决定基于以前的开发经验，把技术进行整合，并组建了游戏引擎的原始开发团队。一个月后，第一版的游戏引擎诞生了，根据其诞生地取名为Los Cocos。一个月后Los Cocos引擎发布了0.1版，其名称从Los Cocos更名为Cocos2D。

团队将研究成果在PyCon 2008和EuroPython 2008上都做了展示，2008年正值乔布斯公布发展iPhone游戏的想法，因此他们决定抓住商机，在2008年6月宣布跟iPhone平台进行接轨，并在当月就公布了用Objective-C编写的Cocos2D for iPhone 0.1版，它与Python版的Cocos2D拥有相同的设计思路。截止到2008年12月，App Store上已有超过40个用Cocos2D引擎开发的游戏。

2009年，团队设计了Cocos2D的世界编辑器



图1 Michael Heald设计的Cocos2D家族LOGO

Cocos2D-Python，这款所见即所得的编辑器使用起来十分方便。此时，Cocos2D的各种平台、各种编程语言移植版也开始涌现， ShinyCocos（Ruby bindings）、Cocos2D-Android（Java based）、cocosNet（Mono based）都诞生了。同时，英国的设计大师Michael Heald为Cocos2D设计了新的LOGO，如图1所示（此前Cocos2D的LOGO是一个奔跑的椰子）。

同时Cocos2D社区的开发者作出了最早的周边工具——Zwoptex和Particle Designer。Particle Designer的功能非常强大，开发者曾需要耗费2个小时制作的效果，在使用Particle Designer后只需10分钟便能完成。随后《StickWars》成为第一款在美国区付费榜夺得冠军的Cocos2D游戏，这宣告了Cocos2D的时代已经到来。

2010年，团队又做出了一个重大的举措：把Cocos2D-iPhone接入到Mac。此举的主要目的是方便世界编辑器的开发，把Cocos2D游戏编译到MAC上只是一个附带的工作，他们原本的想法是创建一个类似于2009年在PyCamp设计的Cocos2D世界编辑器。

Cocos2D-X开启跨平台开发时代

随后，具有历史意义的Cocos2D-X诞生了。X代表着Cross，即交叉。因为Cocos2D-X为开发者提供了跨平台支持，通过C++语言把游戏逻辑一次编写即可编译到iOS、Android以及更多手机平台上运行。

同一时期新西兰人Ryan Williams移植了基于HTML5 Canvas的Cocos2D-JavaScript，社区也出现了Texture Packer、Glyph Designer等丰富的工具和编辑器，以及大量的书籍教程、示例代码、视频培训等供开发者学习和使用。

2011年，Cocos2D家族有了新的发展，集成式的编辑器开始涌现，包括CocoShop、CocosBuilder、Sprite Helper/Level Helper等。另外出现一种利用GCC对Objective-C语言支持来实现Cocos2D-iPhone游戏跨平台编译运行的方式，这种方式需要替换掉Android NDK上默认的GCC编译器。

而在2011年5月，Cocos2D社区发生了一件大事，Zynga雇佣了Cocos2D-iPhone的两位关键作者Ricardo Quesada和Rolando Abarca，两位超级程序员分别从阿根廷和智利迁移到Zynga的旧金山总部工作。

2012年初Cocos2D-X团队再开出两个分支，一个分支是为微软Windows Phone 7移植的Cocos2D-XNA，采用C#语言；另一个分支是得到Google赞助的Cocos2D-HTML5，目前Cocos2D-HTML5还处于早期的开发阶段。

今天的各平台应用商店里，已拥有大量的Cocos2D应用，主要涉及游戏、交互型的电子书、图片等类型。根据官方数据保守估计，目前Cocos2D的总应用量在3500个以上，实际数字应该远远不止。

目前，Cocos2D也拥有了更加稳定的引擎，包括Cocos2D-iPhone、Cocos2D-X，以及被社区普遍看好的Cocos2D-HTML5和JavaScript bindings for Cocos2D-X。

同时也拥有了非常优秀的编辑器（独立编辑器），例如SpriteSheet Editors、Particle Editors 、Font Editors 、 Tilemap Editors。

同时CocosBuilder的集成编辑器也处于开发阶段，这款编辑器目前已能够同时支持Cocos2D-iPhone和Cocos2D-X，而JavaScript的支持也即将进入实用性阶段。

2012年3月，CocosBuilder的作者Viktor Lidholt亦被Zynga从瑞典收编到旧金山总部工作，该软件已经由闭源商业产品改为开源，并将成为Cocos2D社区的官方编辑器。

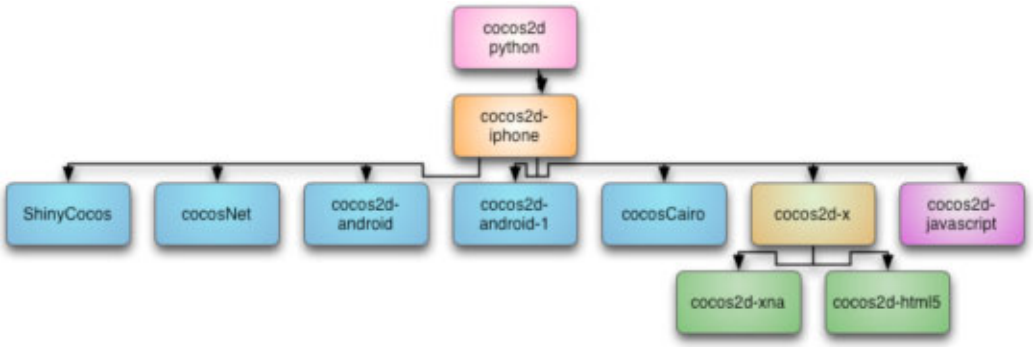


图2 Cocos2D大家族

追求更高效、稳定的开发

虽然现在Cocos2D已有较稳定、便捷的跨平台引擎以及相关的一整套工具，但Ricardo坦言，目前这些引擎还并不完美。他们通过与很多游戏开发者，包括Cocos2D社区的参与者以及游戏开发商沟通，总结出了大家目前的主要需求。

■ **游戏开发周期更短更快。**一方面，开发者希望能拥有更好的编辑器，使开发更加便捷，生成数据的速度更快（比编码更快，因为与编码相比，数据很容易跨平台）；另一方面，Ricardo希望能够为大家提供脚本化的编程语言，例如JavaScript。即使游戏在遇到性能瓶颈时，开发者能够用Objetive-C/C++等原生语言进行开发和优化，然后再绑定出JavaScript接口给性能不敏感的逻辑部分。在多数情况下，直接用JavaScript开发Cocos2D游戏能大幅缩短开发周期。

■ **游戏质量更好。**游戏不会崩溃、没有漏洞以及更加流畅。Ricardo表示，他们会致力于为大家提供更好的编辑器，目标也是做到“less code and less bug”，因为编码更少就意味着Bug更少。

■ **需要原型设计工具。**Ricardo认为利用脚本语言支持+世界编辑器可以让开发者更加迅速地测试自己的想法，如果他们觉得这个想法不好，可以很快地舍弃，同时可以更快地推进他们认为更好的想法。

对于Cocos2D的发展，整个团队也有清晰的规划，2012年的主要计划分以下两块。

第一块是目前他们正在研究的，把JavaScript与现有的Cocos2D-X和Cocos2D-iPhone绑定。他们的目标是对于Cocos2D-X、Cocos2D-iPhone、Cocos2D-HTML5都有同样的JavaScript API。

如图3所示，上层用的是一致的JavaScript API，对于不同平台只需用不同的引擎来实现，对于游戏开发者，在同样的API之上用JavaScript写完游戏逻辑，就可以在iOS、Android以及各种浏览器里面运行，而不必关心底下的Cocos2D引擎具体是如何实现的。

2012年计划的另一块就是CocosBuilder编辑器，他们打算将其做成“所见即所得的编辑器”，可以同时支持Cocos2D-X、Cocos2D-iPhone以及Cocos2D-HTML5。



图3 不同引擎有同样的JavaScript API

Cocos2D-X框架中间层演进

文 / 邹建峰

作者认为Cocos2D能够帮助解决游戏开发时的适配挑战，同时，通过脚本绑定和解释器构成的中间层能为Cocos2D引擎带来更强大的生命力。

游戏开发面临适配挑战

移动设备开发最大的问题就是适配问题，适配又分为两个层面：一个是手机平台（操作系统）层面的适配，另一个是手机机型的适配。在手机平台层面的适配上，游戏开发现在越来越注重游戏的可移植性和多平台化，特别是现在的多个平台鼎立的局面下，游戏开发商不可避免地要开发出多个平台版本的游戏，在开发多平台的游戏中会遇到这样的问题：一是平台非常多；二是每个平台的升级速度非常快。例如：从iOS 3.x升级到4.x、再到5.x，苹果公司每年都升级一个版本，如果不出意外的话，iOS 6将于2012年6月推出。在这个前提下，移动设备发展的速度非常快，平台的变动也非常快，新功能不断涌现。对于较小的游戏开发团队来说，如果一直不断地适应新平台，或者跟着平台升级的话，会是比较痛苦的事情。

在手机机型层面的适配上，给游戏开发商带来的最大的影响就是屏幕分辨率的不同。iPhone 4之后，iPhone的屏幕从HVGA升级到LG 2X的Retina模式，Android平台更有甚者，在我们刚刚组建Android研发团队时，屏幕分辨率只有HVGA，而现在流行的是WVGA，分辨率最高已经达1280x960，测试往往需要购买很多不同分辨率的设备，一个个用于检查，非常头疼。而最新的the new iPad屏幕分辨率已远高于PC屏幕。

因此，游戏开发商在技术选型之前，就要定下来使用哪种客户端技术来进行开发。开发的游戏要实现跨平台发布，目前游戏开发商首选平台是iOS，包括iPhone和iPad设备，其次是Android平台，如果有余力，可能会考虑Windows Phone平台或PC平台（微客户端或Flash网页端）。

Cocos2D-X优缺点同样鲜明

为了减少研发的工作量，使用游戏引擎是必须的，其次要求在游戏引擎之上的编码是一次性的，至少90%以上的代码是一致的。因此源于Cocos2D-iPhone思维的Cocos2D-X引擎就脱颖而出了，在Cocos2D家族中成为最热的一颗明星。

然而Cocos2D-X还是存在一些不足。

- 首先是开发语言的问题，由于iOS的编译器和Android平台发布的NDK都使用C/C++语言，使得Cocos2D-X正好可以实现一次编写代码、编译后就可以在不同的平台上的运行，但这个原则并不适合于不支持C++语言的平台，例如Windows Phone 7平台或Flash平台。为了支持Windows Phone 7平台，Cocos2D-X组织和OpenXLive团队合作研发了Cocos2D-X For XNA的移植版本。在使用这个版本的引擎时，游戏开发商还需要重新将C++转为C#代码，相当于重写了一次，尽管很多代码有其类似之处，工作量仍不可避免。

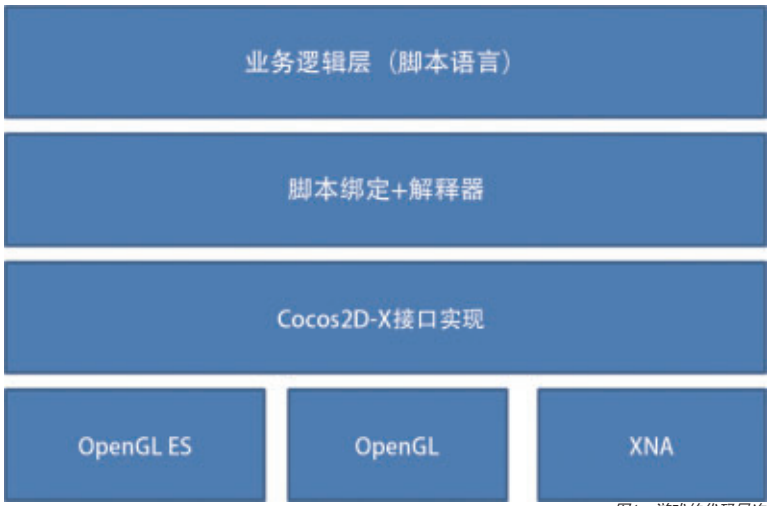


图1　游戏的代码层次

■ 其次是语言特性所带来的招聘问题。移动互联网火爆使相关研发人员短缺，很难招聘到高级研发人员，特别是使用C++语言的开发者，新手也无法通过短期培训而成长为合格的游戏开发人员，特别是由于语言特性带来的细节问题，例如内存泄漏、类型转换等问题都可能给游戏的品质带来较大的影响。C++的内存泄漏虽然可以通过很多工具检测出来，但相对来说没那么容易，而且很多时候这是一件亡羊补牢的工作。

我们在开发第三款游戏时开始尝试Cocos2D-X引擎来研发，Cocos2D-X引擎的优势是很明显的：第一是授权宽松，代码可以进行自由的修改；第二是代码开放；第三是用于验证的产品多、社区相对活跃、Bug修正及时，亦可以提交修正方案。同时，为了弥补中间一些业务经常变动、对效率要求不高的需求，尝试性地使用了脚本语言Lua来作结合，借机也是想尝试一下它能否给Cocos2D-X引入中间层，屏蔽掉具体的游戏引擎。此外，更关注的还是期望使用这个中间层以跨越更多的平台，真正做到“一次开发，全平台运行”的目标。图1是我们游戏产品代码的层次。

中间层带来强大生命力

脚本绑定和解释器构成了一个中间层，Lua的实现已经提交给Cocos2D-X组织，并内置在现在的Cocos2D-X引擎中。在这里，我们称Cocos2D-X不是一个游戏引擎，而更乐意称之为一个接口规范，现在的Cocos2D-X版本可以称为是一个在OpenGL ES的实现。Cocos2D-X组织延用这种思想，在Zynga的帮助下，正在研发绑定JavaScript的Cocos2D-X 2.0版本。

使用中间层之后，有两大优势。

首先是开发效率的提高和入门门槛的降低，根据我们的主观测算，开发效率提高30%～50%。同时，开发人员的入门门槛大大降低，实习生两周后就可以开始写脚本代码，初级程序员只需几天就能上手。由于是解释执行，因此不需要编译和布署，如果脚本编写出错，甚至可以边改边调，无需从头再来，大大提高了开发效率，也不需要关心内存泄漏的问题。

其次是使用的平台更加广泛，我们已提交给Cocos2D-X组织在Mac平台上的实现，也已完成了将整个中间层和在XNA上的实现移植到Windows Phone 7平台上的工作，现在正在进行将中间层和Cocos2D-X接口移植到Flash平台上的工作。这些移植丝毫不会影响产品团队的代码，他们仍继续研发新产品和更新版本，而不需要考虑过多的编程技巧——产品团队只要始终关注产品的趣味性和交互性就可以了，而不是将精力放在代码上。

我们认为，通过这样的中间层，真正才可以将Cocos2D-X引擎转型为生命力更为强大的2D通用引擎，使得游戏开发商将重心放在产品和游戏的品质上，越来越简化研发工作，同时通过更多的辅助工具来实现更多的特性，例如动画播放、布局工具等。因此，Cocos2D-X引擎可以扩大其影响力，成为全平台的2D游戏引擎。



邹建峰

网龙博远无线产品总监、91熊猫看书产品经理，多款手机网络游戏（《战国天下》、《哼哈三国》等）的游戏制作人。

责任编辑：陈博（chenbo@csdn.net）

跨平台游戏开发利器

Cocos2D-HTML5开源2D游戏引擎介绍

文 / 林顺

Cocos2D-HTML5是基于HTML5规范集的Cocos2D引擎的分支，于2012年5月发布。Cocos2D-HTML5的作者林顺将在本文中介绍Cocos2D-HTML5的框架、API、跨平台能力以及强大的性能。

Cocos2D-HTML5是Cocos2D系列引擎随着互联网技术演进而产生的一个分支。该分支基于HTML5规范集，目标是可对游戏进行跨平台部署。Cocos2D-HTML5采用MIT开源协议，设计上保持Cocos2D家族的传统架构，并可联合Cocos2D-X JavaScript-binding接口，最大程度地实现游戏代码在不同平台上的复用。

Cocos2D-HTML5由Cocos2D-X核心团队主导开发和维护，行业领袖、HTML5大力推动者Google为这个项目提供支持。Zynga、Google等大公司的工程师也参与到Cocos2D-HTML5的设计工作中，在各路高手的协作之下，Cocos2D-HTML5的技术方案和API设计正毫无疑问地朝着 大气、简洁的方向快速发展。

Cocos2D程序员可分为三类：苦逼程序员，把自己游戏针对iOS、 Android、Windows Phone三大主流平台分别用Objective-C、Java、C#移植一遍，不是熬夜换语言重写游戏逻辑，就是学习不同语言为熬夜移植做准备；普通程序员，采用C++，一次编码后争取最大限度地在iOS、Android、Windows 8 Metro之间复用；而作为后PC时代的文艺程序员，我认为跨全平台才是真的跨，配合Cocos2D-X JavaScript-binding，最终目标是能够实现Native App和Web Browser之间的轻松跨越。

下文就细致地介绍Cocos2D-HTML5引擎以及如何玩转它。

引擎的框架设计

Cocos2D-HTML5引擎框架的设计目标总结下来有两个，第一个目标是 “偷懒”，引擎运行目标平台是所有的平台，无论是PC浏览器还是移动浏览器，；第二个目标还是“偷懒”，兼容Cocos2D系列引擎，让已掌握开发要领的同学可以继续偷懒，让没有学过的同学可以轻松悟透、轻松入门、晋级。

在引擎的框架设计中，渲染层是Canvas或WebGL，如果浏览器支持WebGL，自动优先选择WebGL，运行速度会快一点，要是不支持也没关系，Canvas通过性能优化后，比起WebGL一点也不逊色。

区别于其他Cocos2D系列引擎设计，Cocos2D-

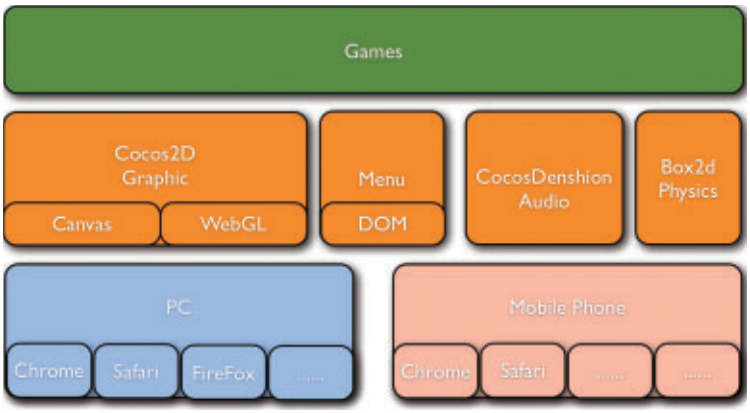


图1 Cocos2D-HTML5游戏框架结构

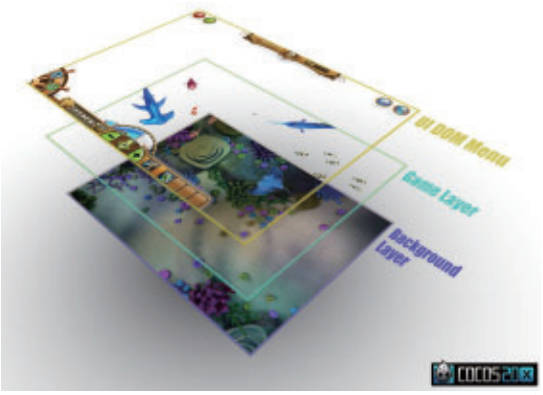


图2 UI Dom分层结构

HTML5引擎框架引入了Dom Menu的设计，浏览器支持多语言的优势得到了传承，开发者再也不用为游戏的多语言发愁了，再也不需要辛苦地到处找字库、贴图了，因为在引擎内已准备好了。值得注意的是，Dom实现的各项菜单还有各种动作和特效，甚至和Canvas可以实现互动，让开发者一点都感觉不到Dom的存在。

引擎的API封装和使用介绍

框架设计得简洁明了，API当然也不能复杂，Cocos2D-HTML5在API的封装上，直观易懂，最大程度地保留原Cocos2D系列的API的使用习惯，保持发扬偷懒的风尚，使开发者可以在悟解框架之后轻松继续晋级。

在引擎的设计和开发中，采用的是John Resige类的继承方法，在使用上类似于C++，适合原有Cocos2D系列的游戏资源进行快速1:1移植，我们的目标是实现机械式翻译，因此期待自动翻译的工具出现。

当然，Cocos2D-HTML5引擎面向的是Web，引擎在高级API的封装上，根据JavaScript语言的特点重新进行封装，满足广大前端程序员、JavaScript开发者的使用习惯：Write less，do more。

API足够容易使用。在其代码仓库有专门的目录，其中为开发者准备了各种Demo，所有的引擎功能都有对应的test case展示其用法，开发者可以在这个基础上快速掌握各种功能和特效的使用，还有更简单的方式——直接拷贝组合应用于游戏开发中。

此外，引擎采用原生的JavaScript语言开发，可兼容其他成熟的JavaScript框架，开发者可以在引擎的基础上自由选用合适的框架，加速游戏的开发进度。

引擎的跨平台能力

在互联网领域，各个浏览器之间存在大量的标准和不兼容，引擎一定要将这种耗费无谓青春的行为扼杀。在引擎的设计中，通过浏览器能力检测和适配器模式，消除不同浏览器带来的运行环境的差异，兼容不同浏览器的事件处理，抹平不同浏览器之间的各个坑，让开发者能在平坦的道路上加速前进，快乐写代码。抹平了差异，再也不需要做重复无价值的工作，基于Cocos2D-HTML5开发的游戏终于可以轻松穿越操作系统运行在各浏览器上了。

采用Cocos2D-HTML5这类更高级语言引擎进行游戏开发，开发效率更高，加上丰富的工具集支持、实时的各种调试手段和云合作开发手段，更加直接的效果调校方式，完全没有开发环境依赖，各种优势的叠加，可以大幅度降低开发成本，提升游戏的上架速度。

如果觉得这样还不够，还有更好的选择，引擎的高级API和Cocos2D-X JavaScript-binding引擎API是相同的，同一份JavaScript游戏代码不用修改或者小范围修改，配合JavaScript-binding引擎就能以Hybrid的方式直接作为本地应用发布。

通过两种发布方式的结合，Cocos2D-HTML5可实现立体覆盖，通过它开发者就能以一个更快、更

惬意的方式实现自己的想法。

引擎的性能

引擎通过一系列的优化措施，在设计之初就注重各项功能的实现和性能的有机结合，保证引擎在使用上没有各种性能上的短板。

目前引擎通过优化，运行起来已符合“复杂”游戏运行的性能要求。但不可否认，引擎采用的是脚本语言，且运行在浏览器这个特殊的环境中，引擎的效率相对于本地应用性能会低一些。

HTML5游戏引擎中，WebGL性能够强，可惜不是所有浏览器都能支持，要实现伟大的跨平台目标，引擎就必须兼容性能次之的Canvas。还好

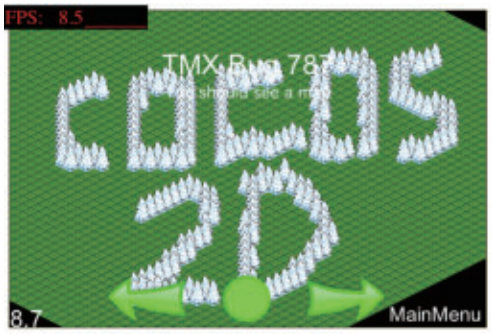


图3　优化之前的TileMap测试例（4400个Tile 8.7帧）

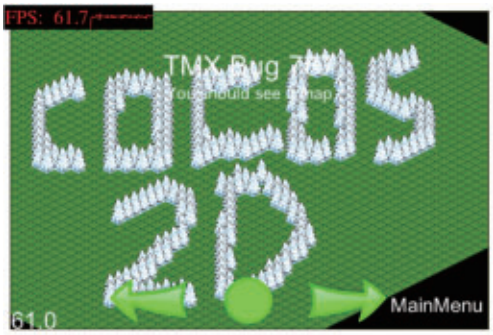


图4　优化之后的TileMap测试例（4400个Tile 60帧）

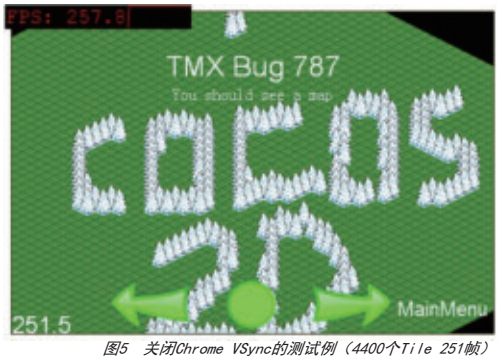


图5　关闭Chrome VSync的测试例（4400个Tile 251帧）

Canvas的表现还不错，经过优化，Canvas模式下引擎性能表现毫不逊色。

各项性能优化工作如图3、图4所示，如有更好的方案请不吝赐教。

首先，从逻辑层面和渲染层面相互配合，引入DirtyRegion的方法，利用少量的逻辑控制，在耗时最大的渲染环节，不该重画的部分坚决不画。通过优化，需要做的事情变少了，渲染效率高了，带来的更大利好是系统级别的省电，在移动设备上，这个优势会变得日益明显。

其次，引擎利用临时Canvas作为缓存，用来缓存需要大量渲染处理的图层中间结果，频繁使用时，直接复制，不需要经过繁琐的步骤获得。通过优化可提升渲染环节的处理效率，加速引擎在特定情景模式下的运行速度。

下面列举一个性能对比的例子：在进行优化之前，运行TileMap测试例，4400个Tile构成的地图，每次在画面中显示，都需要将无数多的Tile拼成一幅图，再显示出来，想想也知道会多么浪费资源。在Mac Pro 13寸上的Chrome浏览器上，Canvas模式下只能运行到8-10帧。引擎通过图层缓存来优化，效果立竿见影，相同环境运行帧率立刻就有60帧了。

那么就只能达到60帧吗？其实不是，只有60帧是因为浏览器有帧率控制。为了看到极限速度，可以将Chrome的帧率限制关闭掉，硬耗CPU计算，可以看到这个测试例的满负载帧率可达250多帧。不过，如此大的帧率已超出显示器的显示范畴，没有实际意义，它只是用来告诉我们它究竟能跑多快。

最后，引擎性能优化工作还体现在不可视区域渲染和像素级渲染的优化上。Canvas默认是画出你要它画的所有东西，是个忠实的执行者，不管你是否画在了不可视区域内，不像OpenGL会把不可视区域自动优化掉。所以我们选用手动模式，优化之后效果也是非常卓越的。

至于像素级的渲染优化，在非拼接类型的Sprite、Label等资源，就可以应用这类型的优化手段了，其原理是避免了Sprite渲染位置处于在两个像素之间，Canvas要去产生边缘各是半个像素的渲染



图6　优化前后的显示效果对比

的情况，提高Canvas的渲染效率，同时还可以获得更好的显示效果。

通过优化前后显示效果的对比，可以看出在优化之后，Canvas只需直接渲染显示原始的资源图片，而不需要做像素级别的调整处理。

图6中，优化前人物裤子上的线本来是单个像素的宽度，结果显示位置刚好在两个像素之间，Canvas要运用虚影、叠加等各种手段在上面2个像素的区域模拟出1个像素的效果，可以说出力不讨好。

通过优化，Canvas可以避免这种严重浪费资源的情况，直接渲染加载图片，提升Canvas的渲染效率，同时获得没有虚影的更佳显示效果。当然，需要拼接的资源不能采用这样的优化方法，防止出现渲染真空地带，但很多类型的游戏，存在大量不需要拼接的资源，有广阔的运用空间。

在Canvas模式下，引擎还优化了Tint的功能，通过图层拆分得到红、绿、蓝和黑色四个基色的图层，然后通过数学方式叠加得到想要的各个颜色效果，避免了用像素级别渲染这种效率极低的方式来实现Tint功能，从而大大提高了Tint功能的执行效率。

当然，这里讨论的都是Canvas模式，用WebGL实现这个功能同样无压力。

在浏览器支持WebGL硬件加速模式下，引擎的压力就小很多了，其渲染性能远远高于Canvas模式，各种优化措施更加齐全，这里就不一一列举了。通过性能优化，引擎渲染环节占用的时间缩到最小，引擎可以支撑更复杂的游戏流畅运行，各种更炫的效果也更多，各位开发者在使用时就不会有如履薄冰的感觉了。

结语

经过各方不懈的努力，现在Cocos2D-HTML5引擎终于踏出了坚实的第一步，目前已经发布了Alpha版本。

可能很多人还在抱臂观望HTML5的游戏发展，而Cocos2D-HTML5游戏引擎已经出发了，我们会将《捕鱼达人》游戏移植到Cocos2D-HTML5，让这个游戏来进一步趟平浏览器上各种各样的坑和引擎的Bug。

时间之梭没有停止，各类型设备计算能力正飞速提升，运行性能上的约束会越来越小，Cocos2D-HTML5凭借其在开发便捷、成本低和发布渠道多样化的优势，会获得越来越大的应用空间，变成游戏开发者优先考虑的游戏引擎，很快就有这么一天到来，不管你信不信，反正我们是.相信了，并在为之努力。

Cocos2D-HTML5项目还处于初期阶段，如果开发者在使用时遇到问题，欢迎登录www.cocos2d-x.org论坛提意见。作为开源项目的Cocos2D-HTML5要想走得更远，离不开大家的帮助。



林顺

开源引擎Cocos2D-X的共同创立者和核心开发者，目前负责Cocos2D-HTML5开源引擎项目的开发，致力于构造一个跨多平台和跨浏览器的游戏框架。

责任编辑：陈博（chenbo@csdn.net）

Cocos2D-X for XNA游戏开发指南（上）

文 / 秦春林

本文介绍了XNA版本相对于C++版本中CCApplication、CCNode的变化，绘制、管道、资源等注意事项以及对集合和数组的处理。下期文章将通过一个开源的Cocos2D游戏展示简单游戏的开发过程。

Windows Phone应用开发

我们知道，在Windows Phone系统上开发应用程序有两种选择：Silverlight for Windows Phone和XNA Framework。

Silverlight是一个基于XAML的事件驱动的应用程序框架，XAML是一种描述Silverlight界面组件的标记语言，利用Expression Blend工具可以设计出非常绚丽的应用程序外观。Silverlight集成了一些传统的控件，实现高品质的文本、矢量图、媒体动画效果及数据绑定等功能。Silverlight适用于开发具有固定页面外观的应用程序，如新浪微博。

XNA Framework其设计初衷是为了让游戏开发变得更简单。使用XNA，一方面你不必考虑创建窗口以及管理它，你需要关注的就是你的游戏逻辑，XNA提供了图形设备访问的接口；另一方面XNA对于游戏中重要资源的管理做了统一的优化处理，内容管道将会为你处理内容的导入、编译和载入。

同时，XNA也是一个跨平台的游戏框架，你可以轻松地移植游戏到Xbox360、PC和Windows Phone设备上。本文，我们主要讨论基于游戏轮询的XNA游戏开发。

Cocos2D进入Windows Phone平台

我们看到，XNA已帮助我们处理了很多事情，大大降低了游戏开发难度，例如不需要管理窗口、统一的内容模型以及使用托管的C#语言开发游戏。然而在游戏逻辑之外，XNA只提供了一些基本的编程接口，我们还需要维护游戏界面各种元素的层级关系，实现精灵在游戏中的各种炫酷的动作，甚至创建一个复杂的地图、使用粒子效果等。这些都是一个游戏引擎实现的功能，然而Windows Phone平台一直缺乏较好的游戏引擎。

在iOS平台，Cocos2D是一个公认的优秀2D开源游戏引擎。在各大移动终端百花齐放的时代，跨平台已成为游戏厂商重要的关注点，Cocos2D-X正是基于Coco2D-iPhone采用C++移植的版本，支持包括iOS、Android、bada、BlackBerry Playbook、Windows XP、Windows 7、Linux等多个平台。这使游戏开发者能够以同一套API、甚至同一套代码将游戏分发至多个游戏平台，大大减少了多个平台之间的研发和维护成本。目前已经有200多款使用Cocos2D-X实现跨平台的游戏，其中不乏Zynga、Glu、Disney Mobile、空中网、网龙、Chillingo各手游大厂的作品和大量荣登App Store Top10的中小团队乃至个人开发者作

品，去年以来国内大热的《捕鱼达人》也正是基于Cocos2D-X引擎开发。

正是在此背景下，Cocos2D-X社区联合国内知名的Windows Phone社交游戏平台OpenXLive移植了Coco2D-X for XNA版本，大大降低了在Windows Phone平台上开发游戏的难度。

Cocos2D-X for XNA概述

2012年2月17日，Cocos2D-X for XNA发布了第一个Beta版本，目前已更新到0.1.1的Release版。

在移植过程中，XNA版本不可避免地会发生较大的变化，因为它不仅仅是将C++翻译为C#，底层的绘制接口也由OpenGL变换为XNA。即便如此，在翻译过程中我们还是保持了大部分接口一致，以至于大部分的代码都可以由Cocos2D-X轻松1:1转换为Cocos2D-X for XNA。

相信大家也看过不少Cocos2D-X引擎相关的教程文档，知道Cocos2D引擎各个版本保持接口一致，这里我就不重点介绍引擎结构，而结合移植过程给大家分享一些Cocos2D-X在XNA版本的变化及注意事项。后面我会以一个实例带大家体验Cocos2D-X for XNA快速开发一个简单的小游戏。

下面将沿着主要类的变化来分析XNA版本的变化，在每个变化的类里，将说明一些变化以及为什么会有这样的变化，主要涉及的类包括：

- CCApplication
- CCNode
- CCSprite
- CCSpriteBatchNode
- CCImage
- CCRenderTexture
- CCSAXParser
- Zip Support

CCApplication的变化

关于CCApplication，主要有以下三方面的变化。

第一个变化是因为XNA中去掉了窗口管理，所以去掉了CCEGLView相关的类。

第二个变化是游戏轮询结构的变化。在XNA中，CCApplication继承于DrawableGameComponent，所以去掉了run方法中实现的循环，因为GameComponent本身就实现了游戏的循环结构。CCApplication的Update重载方法处理Touch事件，Draw重载方法调用CCDirector的mainLoop方法进入游戏递归绘制元素。

第三个比较重要的变化是，CCApplication增加了一些XNA绘制相关的全局属性，包括：

- ContentManager：用于管理资源的加载；
- SpriteBatch：用于绘制2D的纹理；
- WorldMatrix：当前着色器的世界矩阵；
- ViewMatrix：当前着色器的视图矩阵；
- ProjectionMatrix：当前着色器的投影矩阵；
- basicEffect：用来绘制元素的着色器。

这些属性主要用于绘制，它们会被CCSprite、CCSpriteBatchNode以及CCTextureAtlas等与绘制相关的类使用。如果游戏中需要自己绘制2D或者3D的元素，也可以直接调用这些属性进行绘制。

CCNode的变化

在Cocos2D中，CCNode提供了Cocos2D中各个元素的一些公共属性、节点的层级结构以及主要实现元素递归绘制的逻辑。由于Cocos2D中每个节点的位置和变换是相对于父节点的，因此CCNode还承担重要责任，那就是调整着色器的世界矩阵变换。

在OpenGL中，有一个当前矩阵的概念，并且有一个矩阵栈，当开始绘制CCNode时，将当前矩阵压入栈，然后根据当前CCNode的位置、缩放、旋转、扭曲等属性进行一个变换计算，然后将当前矩阵左乘这个变换，就可以实现相对父元素绘制节点。

然后在XNA中没有当前矩阵的概念，因此需要模拟一个这样的概念，于是在XNA版本中做了一点变化。

只不过这里并没有实现一个矩阵栈，因为在这里的矩阵栈都是通过一些列的左乘形成的，只要按相反的顺序左乘对应矩阵的逆矩阵，就可以返回

栈中的矩阵。所以这里添加一个Matrix类型的变量m_tCCNodeTransform来保存每个CCNode的变换，然后在transform方法的末尾，将着色器的世界矩阵左乘以该矩阵，以将着色器变换到当前元素相对于父元素的位置：

```
app.basicEffect.World = m_tCCNodeTransform * app.basicEffect.World;
```

在visit方法的末尾，当前元素绘制完毕时，将着色器的世界矩阵左乘该矩阵的逆矩阵，以将着色器变换回到上一级元素的位置：

```
asicEffect.World = Matrix.Invert(m_tCCNodeTransform) * CCApplication.sharedApplication().basicEffect.World;
```

除此之外，CCNode没有其他变化。

XNA中的绘制

在XNA中绘制3D（注意，Cocos2D虽然是2D引擎，但底层是采用3D绘制的），归纳起来有以下六个步骤：

- 构造一个basicEffect，这在CCApplication的初始化中构造；
- 设置摄像机的视图矩阵和投影矩阵，这在CCDirector中的Projection属性赋值时设置；
- 给basicEffect的Texture属性赋值一个纹理Texture2D；
- 构造顶点数组VertexPositionColorTexture，描述每个点的位置，颜色等信息；
- 调用DrawUserIndexedPrimitives或相应的方法进行绘制；
- 绘制的部分，几乎都体现在CCSprite.Draw方法中，当然在Draw方法中还涉及到许多参数的设置，比如混合因子、透明度等。

以上几乎是Cocos2D-X for XNA所有绘制的部分，所不同的是在CCSpriteBatchNode和CCTextureAtlas中是使用一个纹理同时绘制多个Sprite。

而所有跟绘制相关的变化，基本上都封装在Draw方法中，比如将Cocos2D中的Quads数组转化为VertexPositionCOlorTexture数组，Cocos2D中的混合因子转化为XNA中的混合因子，这样使得其他方法基本上不发生变化。

关于管道和资源

在Win32或者大多数开发场景中，资源（声音、音效、图片、3D模型等）一般都是以文件流的形式在运行时加载到程序中，然后由一些内容导入器对特定格式的文件进行解析处理，系统往往提供一些受支持的格式，另外一些特殊或自定义格式文件则需要开发者自行处理。并且不同的平台处理内容的方式还不太一样，例如在Xbox 360和Windows Phone上对声音的支持都不一样。

在XNA中，为了简化游戏内容的加载，这部分被抽象成内容模型（Content Pipeline）。当你把文件添加到XNA Studio项目中时，它们自动被处理，编译成适合当前选定平台的输出格式。不过这样比较麻烦的是它要求每一个原始内容文件格式都要被支持。

由于Cocos2D主要是由iPhone版本发展而来的，它支持的许多都是iOS中常用的格式，如.plist、.tmx等，这些格式的文件在XNA中是原生不受支持的，因此需要实现自定义的管道用于编译和加载这些类型的文件。

所以在Cocos2D-X for XNA中添加了一个Cocos2d.Content.Pipeline.Importers的工程，用来处理这些格式的文件。但是这里的处理方法比较简单，仅仅是直接转化成文本，所有的解析都需要在运行时完成。

如图1，我们看到选中了自己的导入器和处理器TextImporter和TextProcessor。因此在开发者自己添加的plist或者tmx等格式中，都需要按照tests项目中该类文件一样的设置，以保证能够被引擎正确解析。

不过，这种处理方式并不标准，它只能用于引擎内部使用，如果开发者使用自定义的plist文件，那么不得不在程序中自己去解析，就像引擎所做的那样，所以后续我们也会考虑在管道里直接做

得通用一些，因为像自定义plist的文件使用非常频繁。

另一点需要注意的是，在win32中，.fnt、.plist、.tmx等格式文件中对应的图片资源基本上都是和该描述文件处于同一目录，但在XNA中，资源的识别不包括文件名扩展，所以这三类描述文件对应的图片资源都需要放在同级目录的images文件夹下才能被引擎正确解析。

关于集合和数组

Cocos2D中实现了对集合和数组的处理，例如CCArray和CCSet。其实对于其他一些类型如CCPoint、CCDictionary、CCSet等在Cocos2D中也有自己的封装，这样做的好处是在不同平台对上层的接口保持一致，同时避免在不同平台上这些类型的实现差异，将这些差异隐藏在引擎内部而使接口保持一致。

尽管如此，当初在移植时，我们还是选择用C#本身的Array和List类型代替Cocos2D中的CCArray和CCSet，这样做的目的是为了赶进度，然而有很多开发者反映对这些封装的改变会导致移植上需要改变很多地方，所以后期我们还是尽量改回Cocos2D原生的封装类型。

然而这里需要强调的是开发者自己在移植过程中需要注意的地方，这也是Cocos2D-X for XNA引擎在移植过程中遇到的比较大的问题，那就是关于C++在对数组上应用指针来索引。

例如我们看看C++中初始化数组的方法：

```
m_pVertices = malloc((m_sGridSize.x+1) * (m_sGridSize.y+1) * sizeof(ccVertex3F));
```

在C#中对应则是：

```
m_pVertices = new ccVertex3F[(m_sGridSize.x + 1) * (m_sGridSize.y + 1)];
```

在C++中我们可以通过指针定位到数组的不同位置，如果知道某个指针的数据类型，可以转化为对应的类型进行使用，同一个指针可以转化为不同类型的数组。而C#中则不能使用指针（在Windows Phone中不能使用不安全代码），这些

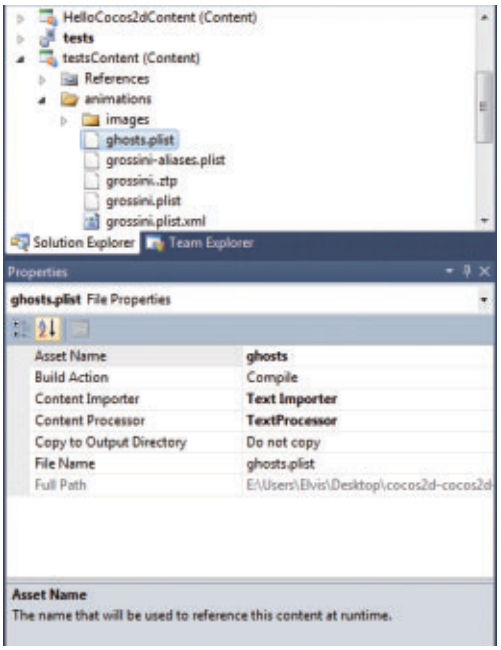


图1 Cocos2D-x for XNA中的导入器和处理器

通过指针索引和操作的方式，全部需要改为基于数组索引和使用强类型，所以这会导致移植上的一些工作量。

小结

本文我们总结了一些Cocos2D-X for XNA版本中比较大的变化和注意事项，剩下还有其他一些小的细节，比如CCSAParser中采用XMLElement流式读取文档的方式解析数据，对于gzip和zlib的支持采用了SharpZipLib和zlib两个类库，同时将get和set方法改为C#中的属性等，大家可以慢慢去发现。下期将通过一个开源的Cocos2D游戏展示简单游戏的开发过程。



秦春林

快乐技术沙龙发起者，OpenXLive程序员，Cocos2d-X for XNA开发者。

责任编辑：陈博（chenbo@csdn.net）

Siri详解之执行和服务系统

文 / 张俊林

在4月刊《程序员》刊登的《Siri详解之输入系统和活跃本体》一文中，作者从技术层面详解Siri，主要讲述了输入系统和活跃本体两部分内容。本文将继续深入Siri的技术层面，阐释执行系统、服务系统和输出系统三部分内容。

执行系统

执行系统是Siri系统中最有技术含量的部分，《Siri详解之输入系统和活跃本体》一文有述：“活跃本体”是根据用户的输入信息，将各种词典资源、模型资源实例化然后进行具体加工的场所，而真正的加工过程是由执行系统进行的。执行系统不仅将用户原始的文本输入解析为内部的语义表示，而且要在用户和Siri交互的过程中（多轮会话）决定下一句Siri应该说什么内容，可见其重要性。

执行系统又可以细分为三个主要部件：语言解释器、会话流控制器和任务流控制器。它们之间分工有异，同时又密切合作，一起发挥作用。语言解释器将用户输入字符串解析为语义表示作为输出，而这个语义表示又会作为会话流控制器的输入，会话流控制器根据当前语句所表达的含义，协同任务流控制器一起决定Siri下一步应该做什么或说什么。

语言解释器

语言解释器是Siri中最重要的自然语言处理工具，主要用来对文本形式的用户输入进行解析，将其映射为概念本体层级的信息表示，即理解语言真正的含义。此外，语言解释器也被输入系统用来对用户输入进行提示或者输入补充，而且对语音识别结果后处理也有很大帮助。

语言解释器有以下一些主要功能。

- 解析一切有助于理解用户意图的输入信息，包括用户主动输入信息和设备自动获取信息。信息包括用户输入的单词序列、用户输入时的手势信息、当前对话流的上下文和实时捕获的动态信息（时间、地点等）。
- 句子句法结构解析：使用句法分析器解析句子结构，句法分析的目的是将单词短语等句子构成成分之间的修饰关系分析出来。
- 句子语义解析：将用户输入信息映射为概念本体（ontology）、概念的属性、概念之间的关系等深层次语义信息。在Siri内部，就是指词汇表或领域知识库内概念的实例。
- 消除句法和语义歧义：句法分析和语义分析阶段的结果往往不唯一，这是自然语言本身具有的歧义造成的。语言解释器有助于对上述两个阶段产生的多种可能解析结果进行歧义消除，确定真正正确的解释。
- 在用户输入阶段判断当前用户已经输入的信息是否在语法或者语义上有意义。
- 在语音识别阶段，对于Nuance公司提供的多个可能识别结果进行句法和语义分析，有助于提高语音识别准确率。

假设用户语音输入：Who is playing this weekend

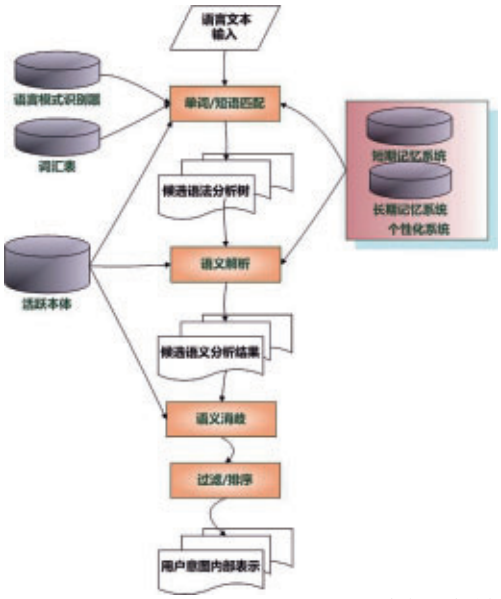


图1 语言解释器运行流程

at the filmore?（本周末谁会在filmore进行演出？）我们来看下Siri是如何对句子进行解释的（如图1所示）。句子经过语音识别后显示在客户端屏幕上，此时，语言解释器以此作为输入开始对句子进行解析。首先，Siri会捕捉上下文环境信息，即当时的时间和地点。然后进入单词/短语匹配阶段。在此阶段，语言解释器会根据语言模式识别器和词汇表将表层单词或者短语映射到概念层。在这个例子里，会将“playing”和“事件场所”连接起来，根据当前时间可以比较明确地确定短语“this weekend”所指的时间，单词“filmore”被识别为“场所名称”。但因为语言的模糊性，这个步骤生成的候选结果可能有多种合理解释，比如将“playing”解释为体育运动领域也是合理的。在这个阶段，个性化信息比如短期记忆和长期记忆也会被用来消解这种歧义现象。在语义解析阶段，Siri会根据句法分析结果来判断语义的一致性程度（和领域模型及相关知识库）。比如这个例子中，“playing”会被解析为“某个场所发生的当地事件”，“filmore”会在领域实体数据库找到，并被解析为“场所名称”。在语义消歧阶段，Siri会考察语义解析的证据强度，比如“playing”被解析为“某个场所发生的当地事件”，“filmore”被解析为“场所名称”，则其语义匹配程度就比只将“playing”解析为体育运动证据强，因为“filmore”并没有体育运动场所的含义。这些匹配活动都是在活跃本体完成的，高强度证据体现为：活跃本体内的概念实体之间有较多的节点，这些节点之间有边能够将其联系起来。经过进一步的“过滤排序”就可以输出其内部语义或意图表示。

对话流控制器

对话流控制系统是在用户的文本表示解析为内部用户意图之后发挥作用，即语言解释器将解析结果传递给对话流控制器，其是语言解释器的后续处理步骤；而任务流控制器则被对话流控制器调用，共同确定Siri下一步应该做什么或者说什么。

对话流控制器有以下几个主要功能。

- 在接收到语言解释器的解析结果后，识别用户意图所属任务类型，即判断用户想要干什么，比如是否要找一家餐馆。
- 对于确定的任务类型或者问题，在获取了语言解释器给出意图的内部表示后，判别执行这个任务需要哪些参数，比如预订或者查找餐馆，则需要确定任务发生的时间、地点以及餐馆属于哪个菜系等参数。假设用户发出的是：“给我找一个附近受欢迎的川菜馆”，那么其中“附近”、“受欢迎”、“川菜”就是“查找餐馆”这个任务所需的参数，其中“附近”是地理条件参数，“川菜”是对餐馆菜系进行限定的参数，而“受欢迎”是一种他人评价参数。在Siri内部，这些参数被称作约束条件。
- 根据当前对话上下文，确定还缺乏哪些必要的约束条件并利用用户的个性化信息（长期记忆以及短期记忆系统），决定应该引导用户说什么，其引导目标就是诱导用户说出Siri完成一定任务所需要的约束条件。假设用户发出请求“Book me a table for dinner”，很明显这个请求过于宽泛，约束不足，于是Siri会问用户：For what time? And how many people?通过这种方式可以获得必要的任务约束条件，便于任务的正确执行。

图2是对话流控制过程的流程图，在获得语言解释器的用户意图内部表示后，对话流控制器会判断意图是否过于模糊或者宽泛。如果是这样，则向用户询问更多的约束条件，否则根据领域模型

和任务流控制器中的信息解析用户意图，并由此确定对话流和任务流下一步应该进行哪些操作或者说什么内容。

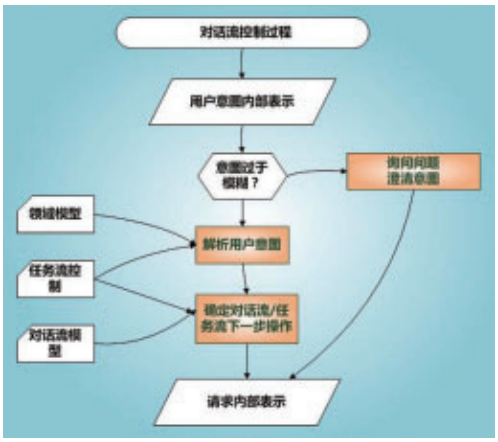


图2　对话流控制过程

对话流控制器在运行时会使用对话流模型，它是对话流控制器的数据提供者，记载了对话流包含的步骤。比如，对于一个高抽象级别的通用任务“执行某项事物”，其步骤应该是：为执行事务收集必要数据，提交事务前确认必需的参数是否充足等。

任务流控制器

任务流控制器的主要功能是界定完成一件任务或者解决某个问题由哪些步骤构成，这些步骤之间是何种关系。任务流控制器和对话流控制器很容易混淆，不容易区分其功能差异。一般来说，对话流控制器主要决定Siri接下来要说的内容或要做的事件，主要是根据领域判断诱导用户提供所需的参数；而任务流控制器更侧重于事务本身的定义，比如一个任务可以切分成若干子任务，是否有时序依赖关系。举个例子：预订餐馆是Siri的一个典型任务，而又可以将其细分成按以下顺序执行的几个子任务：首先要找到一个合适的餐馆；然后检查是否有空位；之后执行预订行为。任务流控制器被对话流控制器调用，因为任务的步骤也是决定了Siri下一步行为或者所说内容的重要组成部分。

任务流控制在Siri中也起到举足轻重的地位，Siri的任务模型由一些领域无关的通用任务模型和若干领域相关任务构成。通用任务是完成一项任务的抽象表述，与具体领域无关，因为其通用性，也可以应用在各个具体应用领域。比如某个事务性通用任务由以下三个步骤构成：首先确定需要事先收集哪些数据，然后执行事务，最后输出结果。这是一项相当抽象的任务，也可以应用在比如订餐、购物等具体领域中。

“受限选择”是Siri中非常常用的通用型任务：其目标是从一个可选择集合中选择出满足一定约束条件的选项。下面我们详述其功能定义及其实现方法。

受限选择任务流

“受限选择”是一种通用任务模型，图3列出了其关联的处理步骤及后续任务模型。所谓“受限选择”，可以理解为根据一定条件从候选列表里进行选择的过程，这里的限制就是一系列约束条件。Siri先根据一定约束条件筛选出满足约束的结果，将结果展示给用户，如果用户找到感兴趣的内容，则可以进入后续的任务流程，否则说明约束条件不足，向用户征求更多约束条件来缩小选择范围。

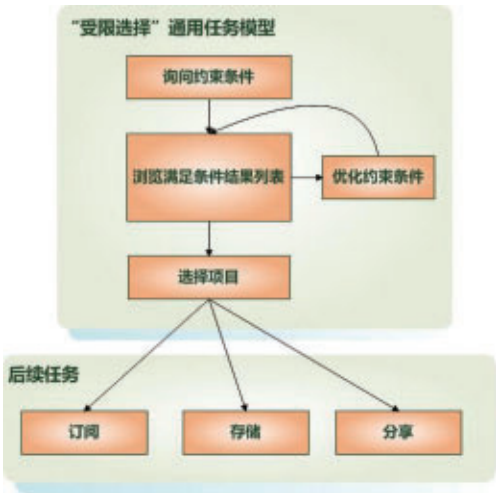


图3　“受限选择”通用任务模型

Siri在实现受限选择任务时，会采用语义转写与信息提示等技术来进行配合。比如用户输入的是：Chinese food in Menlo park。在Siri内部会有如下解析结果：

任务类型=受限选择
领域类别=餐馆
约束条件：

地点=Menlo park,CA

菜系=Chinese food

如果经过对服务的查询发现不存在能够满足约束的服务，Siri会将用户请求语义转写为：“Sorry, I can’t find any Chinese food near Menlo park”。并进一步给用户提示：“you should try other cuisines or location”。

服务系统

Siri本质上是服务导向的用户意图识别系统。无论是对话流控制也好，任务流控制也好，其根本目的还是将用户引导到Siri能够提供的某项具体服务，以达到帮助用户完成某些任务或者解决一些问题的目的。目前Siri可以提供多种领域的服务，这里面涉及到服务管理的问题，即如何进行管理才能使系统可用性高、可维护性强等。具体而言，Siri中有三个子部分有服务功能：服务模块、服务能力模型和服务整合模块。

服务模块

服务模块（service component）登记了Siri能够直接调用的所有服务功能。具体而言，包含以下三种功能。

- 对网络提供服务提供API调用接口。对用户而言，这种网络服务本来是直接通过网络界面可操作的，现在则可以通过API接口由Siri来代为执行。比如，调用API可以向指定的某个商业实体反馈评价信息等。
- 对客户端应用提供API调用接口。比如，客户端可以提供闹钟、发送电子邮件、日历安排等功能，用户原本可以通过客户端提供的操作界面直接操作这些应用，而通过API接口，Siri可以直接通过程序调用来完成一些功能。
- 对一些用户不可见的内部功能的API接口调用。上述两类是用户本身可直接通过操作界面完成的服务功能，但也有些功能是用户无法感知或直接使用的，而Siri可以为其定制API接口，以为其他服务提供数据或功能支持，比如自动获取地理位置信息等。

服务能力模型

服务能力模型（service capability model）记载了调用服务所需的机器可读信息及服务功能和API服务的映射关系。服务能力模型可以管理哪些服务能够回答什么样的查询、哪些服务能进行什么样的事务及这些服务暴露了哪些接口参数等。

服务整合模块

服务整合模块（service orchestration component）是与服务有关的模块中最核心的部分，它调用另两个服务模块为用户提供最终服务内容。完成用户的某项需求通常要调用分布在各处的多项服务，每项服务能够提供部分信息，且服务间有些顺序需要遵守，所以如何调用所需的多种功能，调用顺序如何确定以及如何根据部分信息拼合成最终用户所需服务是其核心内容。具体而言，服务整合模块提供以下功能。

- 动态决定哪些服务能够满足用户请求。
- 动态地以一定顺序组合多个服务。
- 动态地将任务参数转换为满足API格式要求。
- 动态监控服务是否可用，是否好用。
- 将多个服务最终结果整合后规范输出格式。

更重要的功能是对多项服务功能进行分配调度和整合。图4展示了服务整合的流程。

我们以一个具体的例子来解释Siri的服务整合过程（如图4所示）。假设用户提出的需求是：希望找到一家环境比较好的吃晚餐的地点。通过前面若干步骤，Siri将用户需求理解为：找到一个餐馆，这个餐馆就在附近，而且大众点评评价较

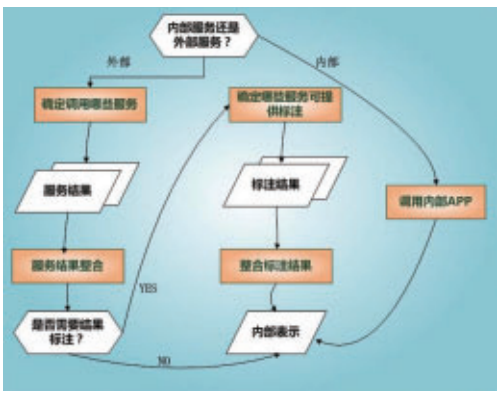


图4 Siri的服务整合过程

高，在一个时间段内有空位，并且满足一定的菜系风味。这些作为服务整合的输入参数。

Siri首先会判断满足这个需求的服务是外部服务还是内部服务。如果是内部服务（比如回答时间信息等），那么直接本机执行即可；如果是外部服务，则需要查找服务模块和服务能力模型来判断哪些外部API能够满足需求，即需要调用哪些外部服务。在这个例子中，明显需要调用外部服务。Siri通过查询服务能力模型，从服务功能和服务质量多个角度找出需要调用的外部服务，并将内部需求转换为外部API所需的参数形式后调用多个外部服务。就上述例子来讲，Siri发现需要调用若干个服务才能圆满完成本次查询，假设服务1和服务2可以提供附近餐馆的列表；服务3和服务4可以过滤出高质量的餐馆；服务5和服务6可以提供关于餐馆的评论信息，服务7可以提供餐馆的菜品信息；服务8和服务9可以提供在晚餐时间是否有空位的查询。所有这9个服务需要同时被调用才能完成本轮服务。之后，Siri调用外部API获得相应的查询结果，并对返回结果进行进一步验证和整合，比如去除重复结果，整合形成最终结果。

通过本轮查询后，如果Siri发现结果个数太少，说明约束条件过于严格，那么可以放松约束条件再次重复上述过程以获得更多结果。比如，如果发现上述例子结果过少，会把“附近”的定义从N公里放松到更大的范围。

还有一些情况，需要一些辅助数据对结果进行标注，那么Siri会执行类似的流程去获取更多辅助数据并将其和原始结果进行整合，形成最终的服务结果提供给用户。

输出系统

Siri的输出系统会将最终提供的服务结果或会话过程的中间内容展示给用户。具体而言，输出系统的功能为：

- 支持多模态输出，包括电子邮件、短信、GUI和语音输出等多种模式；
- 将输出结果统一成内部表示格式，并根据不同模态规划其界面输出布局；
- 对于语音输出来说，可以提供多种人声选择；
- 根据用户偏好自动选择合适的输出模态；
- 可以根据用户喜好变换输出界面的外观。

结束语

Siri是苹果公司新推出的一种新型人工智能框架，不仅在商业宣传上令人耳目一新，在其技术架构和具体实现上也颇具新意。很多人看到演示后会误以为Siri只是单纯的语音识别系统，通过上述技术描述可看出，事实并非如此简单。语音识别只是整个宏大架构中的一个环节，即使是语音识别系统，也是专门针对Siri本身做过功能优化的，个性化语音识别系统。从本质上讲，可以将Siri归纳为面向垂直领域的用户意图识别系统，所有复杂的技术手段都是为了能够理解用户意图，并最终将用户输入引导到Siri能够提供的某项服务上，这样才能真正帮助用户解决一些实际问题。如果只将其理解为一个无聊时解闷的聊天伴侣的话，显然是有失偏颇的。

尽管Siri最初依附于iPhone平台，但很显然，这种依附性并不强。可以预见，这套系统会不断扩展到更多种硬件类型的智能控制，比如车载控制系统、智能电视控制系统等。当然，作为用户意图理解系统，Siri离尽善尽美还有非常大的差距，还需要相关基础技术不断发展来拓展其实用性。但毫无疑问，Siri开创了一个非常有前景和想象力的智能应用方向，而这个方向无疑是值得很多人去进一步研究探索的。



张俊林

《这就是搜索引擎：核心技术详解》作者，新浪微博研发人员，中科院软件所博士，主要研究方向为自然语言处理、搜索技术、推荐系统及机器学习。

责任编辑：杨爽（yangshuang@csdn.net）

可伸缩性的10年探索
知名网站的技术实现

文 / 林昊

网站需要具有良好的可伸缩性，来应对不断增长的访问量和数据量。本文结合Google、Facebook等知名网站的技术发展历程，阐述了它们在可伸缩性上的技术实现。

《程序员》杂志5月刊的《可伸缩性的10年探索：知名网站的技术发展历程》一文中介绍了一些Alexa排名较前的网站的技术发展历程，本文将结合提及的Google、Facebook、Twitter等网站的技术发展历程，总结它们在可伸缩性、可用性、高性能以及低成本四点上通常采用的技术。

可伸缩

可伸缩分为垂直伸缩和水平伸缩两类，垂直伸缩通过升级机器的硬件来解决问题，水平伸缩通过增加机器来解决问题。不同网站在可伸缩上采用了不同的策略。例如，Google完全依赖水平伸缩来解决问题，而其他网站多数是依赖垂直伸缩来解决数据存储问题。

垂直伸缩要求软件要能在硬件升级时，发挥出硬件的能力，例如可配置的并行数、内存等。硬件的发展速度非常迅猛，网站的机器配置自然也是每年都在升级，因此软件时刻都在被检验是否能垂直伸缩。

水平伸缩主要解决的是如何仅通过增加机器就能解决问题，一般通过应用层和存储层两个层次来解决。

在应用层做到水平伸缩，通常采用的策略是Share Nothing/Stateless，将状态信息放入客户端或存储层（例如用户的会话信息放入Cookie，或放入服务器端的缓存系统）。此时，在访问量上涨后增加机器即可。在应用系统由单台增加到多台组成集群时，需要引入负载均衡，可能是硬件的也可能是软件的。

我们发现，就应用层的结构演变而言，前面提到的Google、Facebook、Twitter等网站在应用层上最后形成的结构几乎完全一样，均为前端Web系统集群 + Services集群。通常到了一定阶段后，前端Web系统和Services又会按照一定的规则来进行拆分，如按业务领域等。可见，其实SOA在大网站中已经落地，而不像企业领域中宣传的那么虚。

存储层实现水平伸缩，难度就比应用层大多了。从Google、Facebook、Twitter等几家网站的发展历程也可看出，解决存储层问题需要花费大量的时间和精力，而且由于业务发展的压力较大，很多网站开始会采用垂直伸缩方式来解决存储层的问题。

实现存储层的可伸缩性，通常采用的方案是单点写（指在某个粒度的单点，对HBase而言，单行数据一定在同一台机器上进行写操作），但读可能是多点。读多点，主要需要考虑不一致的问题。无论读是单点还是多点，数据都会在软件层面做到写多份，策略主要有同步写多份、投票写多份、异步复制最终一致等（例如HDFS、Cassandra、ZooKeeper），采用自动分裂方法来实现数据量增长时的自动伸缩，采用一致性Hash或根据某种规则的自动均衡策略等来实现机器增

减时的相应处理，同时也需要有感知机器增减的方法（例如采用ZooKeeper）。

可以看出，要在存储层上实现可伸缩，技术上的难点很多。这也是为什么大规模网站都不在数据库上做复杂的运算，而只是把其当做一种存储信息的方式来使用。存储过程等基本不会出现，以降低数据库的压力。

除了应用层和存储层需要做到可伸缩外，在设计系统时硬件层面的可伸缩也是需要考虑的。例如，硬件负载设备只能支撑到一定流量，同样也要考虑如何让其能够可伸缩。

除了尽可能做到系统可伸缩外，减少对系统的压力也是缓解系统的可伸缩所面临挑战的方法，例如批量提交、最终一致、YouTube提到的“欺骗”、适当的正确性等策略。

可伸缩性是网站从小到大要经历的第一关挑战，同时随着网站的不断发展，还要不断地做技术改造才能保证网站在每个阶段（如同机房阶段、同城多机房阶段、异地多机房阶段）都具备优秀的可伸缩性。不过，幸运的是不同阶段的可伸缩性方案都已经比较成熟了。

可用性

可伸缩性能保证网站在访问量和数据量不断增长的情况下生存下来，同时也从某种程度上保证了网站的可用性。但除此之外，要保障系统的可用性，还需付出很多努力。

设计高可用性系统时，避免单点是其中最重要的一点，一般采用主备或集群等方法来避免单点。例如，负载均衡设备、MySQL等通常采用主备方式。在BigTable里采取了一种比较特殊的方法来避免Tablet Server的单点，即通过Chubby来感知Tablet Server的健康状况，如发现Tablet Server挂掉了，则由Master将此Tablet Server负责的Tablet迁移到其他的Tablet Server上。

除了避免单点外，降低耦合也是设计高可用性系统时应考虑的重点。通常采取异步化来降低耦合，将非关键逻辑和关键逻辑隔离开。例如，打开天猫的商品详情时，都会有相关商品的推荐。如果这个推荐系统出问题的话，就会导致商品详情也看不到了，因此这里可以异步载推荐系统，通常采用AJAX带超时的方式来实现。

Google、Facebook等网站在总结多年的可用性系统设计经验时，列入了同一条设计原则，即保持简单。简单的方案一般比较容易掌控，而复杂的方案一方面实现难度大，另一方面出现问题时很难排查。

可控性也是各网站强调的重点，可控性意味着一切代码都在掌控之下，并且最好是每个使用到的部分都由专业人员负责（对可用性要求越高，在这点上要求也就越高）。这样在出现问题时才能清楚是哪个地方造成的，并且可以自行排查和解决。如不可控，则意味着一旦出现问题，就得依赖第三方来排查，时间上非常不可控。这也是网站不愿意采用商用软件的重要原因。

编写高质量软件是保障系统高可用性的重要一环，可控性是其基础。Google、Facebook等网站在总结保障高可用的经验中均提到了监控/报警、容错、自我保护等策略。

监控/报警是软件自身能够保障高可用的重要策略，就像是汽车的仪表盘一样，可以告诉你油还剩多少、速度是多少、胎压是否正常等重要信息。对于软件而言，同样需要让外部获取到其运行的状况。例如，Google的软件都会提供一个HTML页面供使用者或开发人员访问（用过Hadoop的人也会发现这个特征）。在这个页面上可通过key/value的方式来获取系统的一些运行指标。对于RPC系统而言，Google会采集所有的正常请求、错误请求以及其消耗时间的分布状况（>0.05s、>0.1s等）。除了监控系统的运行状况外，也提供了其他一些方式以便外部能简单判断系统运行是否正常，例如cURL某页面等。对于不正常的现象，要及时报警，尽可能做到在故障尚未影响到用户时解决掉。

软件的正常运行需要依赖很多外部因素，例如机房、硬件、数据库、服务等，而所有依赖的部分都有可能出现故障（要坚信这点，互联网的特色

是所有小概率事件都会发生）。在设计软件时需要考虑当依赖方出问题时，如何保障软件的可用性，因此一定要做一些容错处理。为了避免机房故障，各网站通常会租用或建设多机房。例如，Google采用IDE硬盘来存储文件，不做RAID，于是采用复制三份的策略来避免硬盘故障导致数据丢失。

软件一般会提供多种功能，有的重要、有的不重要的。而由不重要的功能异常导致重要的功能出现问题，显然不合算，因此在设计软件时需要充分考虑异常隔离，不互相影响。例如在Google的系统设计中会采用Prioritized Request等策略。

James Hamilton的那篇著名的论文《On Designing and Deploying Internet-Scale Services》中提到的Graceful Degradation，即为优雅降级。通常采用的方法是在故障将要出现或出现后，关闭系统的某些功能来降低故障产生的影响。例如，网站上有些操作可能特别耗资源，而这些资源的消耗又可能影响到核心功能，因此一旦出现影响，就可以关闭这些功能保障核心功能可用。降级可以帮助系统临时绕开故障，而产生故障的原因需要在事后排查。

交付具有高可用特征的软件是开发人员的重要职责。而对一个网站而言，软件不是一次性交付的，也不是好几年才升级一次，需要频繁交付，因此维护软件是保障高可用的重要环节。

多数情况下，系统的不可用是由变更造成的，因此如何降低变更对系统可用性造成的影响成为了各网站都关注的重点。Google在发布新产品时通常采取“滚木移石”方法，而Facebook则通常采用Dark Launch方法，降低变更带来的影响。

人工处理系统变更是故障产生的隐患，因此各网站基本都会推荐多种工具来实现系统变更的自动化，例如采用Puppet来实现自动化部署。

除了发布这个重要环节外，处理故障也是维护方面的重要工作。系统总有出现故障时，如何快速处理故障以降低对故障可用性的影响也成为网站一直关注的重点。前面已经讲述了可控性对解决故障的帮助。除可控性外，各网站也在研究其他方法。Facebook采用FBAR来自动处理部分故障，这显然可从某种程度上降低故障产生的影响。

性能

前面提到的可控性同样也是保证性能的重点，只有明确知道调用的每个API及所依赖环境（包括软硬件）的细节原理，才能编写出高性能软件。

从系统结构上来看，为了保障高性能，各大网站都采用了类似的方法。首先是前端Web系统这块，都采用了可编译为机器码的方式，即便Facebook采用的是PHP，其仍然研发了一个可自动转化为C++代码的产品来提升运行效率。

设计系统时，应考虑将没有前后依赖的逻辑并行化处理，或者将大的请求进行拆分。例如，Google会先将所搜索的内容进行分词，然后并行进行索引查询，从而提高响应速度。

基本上各大网站都极度依赖缓存，原因在于，内存的访问速度远快于磁盘。在依赖缓存的场景中，最需要做到的是数据一致性的合理保障。一个典型的场景是数据更新时保障缓存一致性的策略。要将缓存的更新和数据的更新做成一个事务显然有不小的难度，此时网站常采用一个简单的策略来保障，就是先失效缓存，再更新数据，等到下次系统去访问此数据时，才更新到内存。

除了缓存外，可以看到各大网站都采用了CDN，CDN可以让静态文件更靠近用户，便于用户快速获取。而除了让静态文件更靠近用户外，多IDC的建设除了提升可用性外，还可以让动态数据更靠近用户。当然，这在技术上的实现难度会比CDN高很多。

除了结构上的这几点外，技术创新是提升系统性能的重要方法。例如，Google提高了TCP的初始拥塞窗口值等。而要做到技术创新，显然可控性是基础。

成本

随着网站规模的不断扩大，系统的运行和维护成本将会成为公司中支出的重要部分。例如，有数据表明，腾讯每年支付给运营商的费用在总支

出中占比排行第二（2010年为208亿）。网站规模越大，成本控制就越重要（潜台词是在网站规模不是很大时，也许支撑业务快速发展更重要）。例如，性价比是Google设计系统时重要考量的指标。有些网站会采用x元/每千次PV来计算成本。

有些性能优化需要增加成本（如缓存和CDN），而有些性能优化是可以降低成本的（如Google对索引结构的优化），因此性能优化通常也是降低成本的一种方法。网站规模大了以后，规模效应可以让有些性能优化带来的成本降低非常明显。

硬件不断升级，而软件系统层面上又更多的是靠集群来支撑，因此通常很难完全消耗硬件资源，虚拟化就成为一种不错的降低成本的方法。虚拟化具备很好的隔离机制，避免了应用间的互相影响，因此落地的难度不大。

除了依靠虚拟化来降低成本之外，Google采用了自行实现的一种Shared Environment方法来降低成本。Shared Environment可根据不同类型的资源消耗，动态组合（例如分时）部署到同一台机器上，充分利用资源。

如前所述，网站主要是靠可伸缩性存活下来的，因此随着规模的扩大，必然会有大量的机器。比如，Google有上百万台机器，Facebook有几十万台机器。在这么大规模下，自行根据应用特征设计机器，会带来很大的成本下降，因此Google、Facebook都自行设计机器和数据中心。从PUE上可以看出，Google、Facebook自行设计数据中心带来的成本降低非常可观。

各网站的情况不同，应对以上四点挑战所采用的方法不同，每个阶段都有自己适用的解决方案。例如，Google成立初期的主要业务是搜索，主要竞争的是技术，功能次之，而Facebook、eBay等网站的竞争压力主要在业务功能上，因此在成立之初必然会有不同的侧重点。不用想着一开始就把网站做成Google、Facebook等现在的结构，适合自己的就是好的。

很多开发人员在加入规模较大的网站后，会觉得系统结构已经稳定了，没有发展的空间，但从上面各网站的发展历程来看，可以看出网站对技术的要求是在不断演变的。通过观察大网站的发展历程，并结合公司的业务背景、知识结构等来判断其下一步的发展，对个人成长是有很大帮助的。同时可以借此储备一些技术，以把握技术演变时的机会，获得更大的成长。如果开发人员加入了规模尚小的网站，且自身技术储备不错的话，就有机会亲身经历网站从小到大的演变了。但这要看个人如何把握。

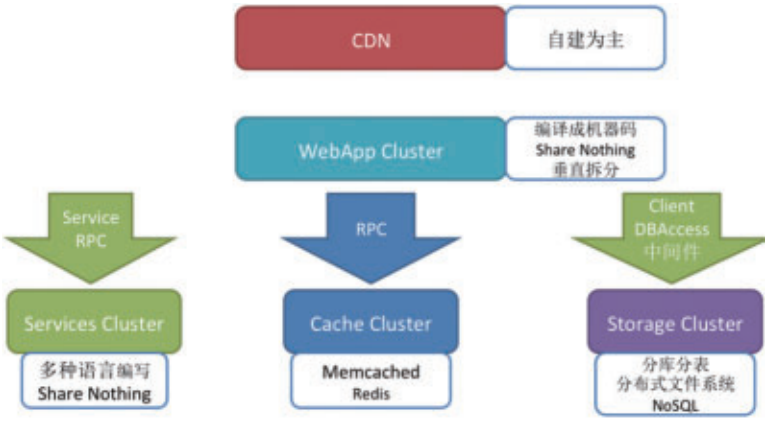


图1　发展到一定规模后的网站结构

围绕可伸缩性、可用性、性能和成本这四个方向，在网站发展到一定规模后，通常会演变成如图1所示的结构。

除了在可伸缩性、可用性、性能和成本这四方面的技术挑战外，网站还面临其他很多方面的挑战，例如海量数据分析和挖掘、网站安全、业务发展的灵活性支撑、人员增长后庞大的软件管理等，因此构建一个支撑大访问量、长期发展、低成本运行的网站是需要有坚实的技术背景作为支撑的。



林昊

目前就职于淘宝，2007-2010年负责设计和实现淘宝的服务框架，此服务框架在淘宝大面积使用，每天承担了150亿+的请求；2011年开始负责HBase在淘宝的落地。

责任编辑：杨爽（yangshuang@csdn.net）

云计算与B2B2C架构实战

文 / 邢波涛

B2B2C是一个基于SaaS架构的平台，可由多个商户共同发布产品。本文用简明的语言分析了典型的B2B2C平台业务需求的架构，并给出了相应的技术实现方法。

B2B2C是Business to Business to Customer的缩写形式。第一个B指广义的卖方（即成品、半成品、材料提供商等），第二个B指交易平台（即提供卖方与买方的联系平台，同时提供优质的附加服务），C指买方。卖方不仅是公司，还包括个人，即一种逻辑上的买卖关系中的卖方。B2B2C定义包括了现存的B2C和C2C平台的商业模式，更加综合化。典型的B2B2C平台是淘宝的天猫网。事实上，传统典型的B2C平台，如京东、当当、卓越亚马逊也都在向B2B2C平台转型。新兴起的B2B2C平台是卖3C水货出名的深圳华强北在线平台。B2B2C平台由于具有天然的招商和分销的性质，所以不仅是互联网背景的公司（例如凡客V+）对B2B2C感兴趣，很多有影响力的传统企业，也想建立自己独有的B2B2C平台，不仅是把商品卖给消费者，还有进一步控制上下游产业链和线下分销商的意思，比如修正药业、海尔、TCL、康佳等。

在架构上，由于B2B2C是多个商户共同发布产品，所以它天然是个基于SaaS架构的平台。而SaaS是云计算的最上层，是云计算的应用层。我反复强调过一个概念：云计算是由“云”和“计算”两个独立含义的名词组成的，“云”并不是“计算”的定语或者修饰语。我理解的“云”，是商业模式，就是告诉别人如何赚钱，也就是SaaS，即“云”＝“SaaS”。可以这么说，如果没有SaaS，那么“云计算”就不能称为“云计算”，只能称为“分布式计算”、“网格计算”或者“并行计算”。一个典型的B2B2C平台业务需求的架构如图1所示。

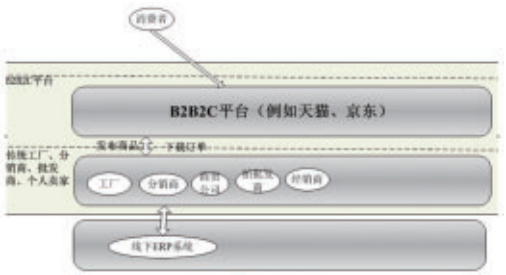


图1　典型的B2B2C平台业务架构

卖家发布商品到B2B2C平台，而消费者，在一个B2B2C平台上（针对消费者来说，他看到的实际上是B2C平台）下了一张订单，则这张订单经B2B2C平台直接路由（如天猫）或者由B2B2C平台的人员审核（如京东）生效后再路由到平台卖家的后台系统上。一个卖家登录到B2B2C平台系统后，可以看到自己的订单，进而选择打印快递单、打印发货单、快递商品到消费者手中，那么这张订单的生命周期也就完成了。大一些的卖家，线下会有自己的ERP系统，他们会通过B2B2C平台提供的API接口，下载订单到ERP系统，冲减实际库存形成财务的收入账款。

根据以上的业务需求，从技术实现上，一个B2B2C平台则需提供如图2所示的功能。

一个典型的B2B2C平台，最基本会提供三个最基本功能：第一，自身是一个B2C网站，有展示商品、为消费者提供下订单的功能；第二，还需要

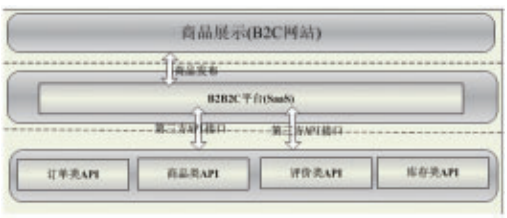


图2　B2B2C技术需求

有与银行或者第三方支付平台进行结算的功能，让消费者完成购物和支付的全过程；第三，需要提供卖家发布商品的功能。实现好上述三个基本功能后，比较成熟的平台（如天猫和京东）还会提供各类API，方便大卖家通过接口和技术的手段，获得信息流。

我的建议是采用最新的HTML5、CSS3样式表和jQuery技术做为表现层，jQuery有很多显示的特效，作为后发者，可以利用最新的技术提高买家的用户体验。虽然国内主流桌面浏览器IE6、IE7和IE8，对HTML5支持得还很不好，但手机浏览器已经全部支持HTML5了，因此，这里可以采用一些既支持HTML5又屏蔽浏览器差异的的框架作为自己的视图层技术。

而对于业务层，我的建议是采用自己最熟悉的技术去做，无论是Java还是.NET。当然我建议还是Java和J2EE，因为无论是负载均衡还是可扩展性，Java都有很多开源的技术实现，以便自己团队挑选最适合的架构。而.NET阵营，由于微软对自身的定位还处在摇摆不定当中，所以最近几年技术路线变化很大，最重要的是前后变化不连贯，背弃了微软一贯向下兼容的优良传统。而Java阵营这方面做得则比较好。即使使用最基本的JSP，也可以运行在最新的Web服务器和最新的JDK版本下。

对于缓存，我的建议是使用Redis。Redis使用起来非常简单，功能强大，与开发语言无关。Java这边，做缓存的老牌技术虽然很多，但即使是O/R Mapping工具本身，例如Hibernate，跟Redis相比较，都不在一个级别。Redis是为互联网而生的，天生具备高并发和高读取，而Hibernate之类的缓存，是为一个具体业务应用、为局域网准备的。B2B2C网站显然是支持高并发的互联网网站。大家都在说缓存，但其实不是一回事，它们的应用场景完全不同。

而对于数据库这一层，对于B2B2C网站来说，单一种类的数据库，估计满足不了需求。我们可以根据业务需求做垂直切分。对于事务要求不高的业务需求，例如登录日志、收藏、评价、网站流量记录，可以选用NoSQL数据库；对于事务要求比较高的业务，利用下订单后和网银的对接，则要

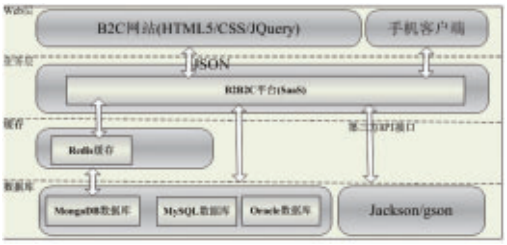


图3　B2B2C技术架构

求对事务支持较好的传统数据库（如Oracle）。对于关系要求比较高的业务，如会员、CRM的分析和统计，则可以使用MySQL数据库。

而SaaS本身的技术架构设计，市面上成熟的理论和实战资料并不多，最常见的就是Salesforce的设计和IBM的几篇文章。而Salesforce的设计本身非常复杂，为了支持以后的扩展，表结构设计也非常复杂，我感觉并不适合开发者作为项目启动使用的设计架构。而IBM的几篇文章，都是跟IBM自己的服务器密切绑定在一起，非得购买IBM的整体解决方案才可以实现。其实，Oracle数据库本身是天然支持SaaS的，对应用层也是透明的，开发者可以像以前一样开发自己的业务系统而无需过度关注SaaS本身。这样的技术架构对程序员和系统架构的要求是简单了，但存在致命弱点：一是只能依赖于Oracle数据库本身，再无迁移数据库的可能；二是即使没有迁移数据库的需求，这类设计对Oracle数据库的DBA要求也极高，而对业务系统的垂直切分，也会非常不利。

在这里，我并没有过多地讨论应用服务器本身的选择。传统应用服务器，无外乎Web Sphere和WebLogic，这类服务器本身比较昂贵，而且也不适合互联网架构，它们同样是为适合企业内部业务系统而设计的。而比较新的一些互联网架构，例如Nginx，我并不熟悉。这篇文章只总结了一些自己的技术心得，要想深入下去，还必须得有实战项目才好。



邢波涛

北京新软孚信息技术有限公司技术负责人。关注SaaS管理软件和B2B、B2C电子商务的融合。

责任编辑：杨爽（yangshuang@csdn.net）

基于P2P网络的分布式消息队列

文 / 高洪

云计算技术的大步发展，使得分布式基础组件备受追捧，而分布式消息队列是其中非常重要的基础组件。本文先介绍了分布式消息队列的关键需求和核心关注点，然后结合实例讲述了基于P2P网络的分布式消息队列。

伴随着云计算技术一日千里的步伐，基于分布式架构的应用层出不穷。相应地，为了解决新问题而产生的分布式基础组件也越来越受到架构师的追捧，其中分布式消息队列就是云计算技术框架中非常热门且有用的一种基础组件。

发展历程

消息队列最初是System V进行进程间通信的一种方式。1993年，IEEE对消息队列进行标准化，使之成为POSIX标准的一部分。之后，POSIX消息队列作为单机范围内进程间的标准通信机制在应用开发中得到了广泛应用。随着分布式技术和云计算的兴起，企业应用开始向大型分布式方向及云计算SaaS模式转变，这些大型分布式应用在开发部署时，又对传统消息队列提出了新的要求。比如，如何实现分布式应用各组件间的解耦，以降低开发和运维成本；如何更方便地实现分布式应用在不同数据中心、不同节点间进行通信，使分布式应用开发者更专注于上层业务流程的构建而无需关注底层通信细节等。其中最重要的一点就是消息队列的分布式特性支持：要求消息队列能够分布在网络中多个节点上，屏蔽底层通信协议，为各种分布式应用提供基础消息通信服务。

在应用需求的驱动下，分布式消息队列得到了空前的发展，支持各种协议和标准的消息队列纷纷涌现。为消除应用和分布式消息队列之间的耦合限制，统一分布式消息队列对外提供的服务，2001年Java标准化组织（JCP）发布了JMS（Java Message Service）标准，对Java平台中分布式消息队列的API接口进行了规范。

目前，JMS已成为业界主流分布式消息队列遵循的标准。与JMS仅规范分布式消息队列的API接口不同，AMQP（Advanced Message Queuing Protocol，高级消息队列协议）是分布式消息队列应用层协议的一个开放标准，它面向消息、队列、路由（包括点对点和发布/订阅）、可靠性、安全等，对消息提供者和客户端的行为进行了强制规定，使得不同厂家的产品真正实现了互操作能力。很多开源分布式消息队列像ZeroMQ、RabbitMQ等都实现了对AMQP协议的支持。

关键需求

除分布式和队列两个基本特性外，分布式消息队列还需要关注基本消息模型、功能和性能、接口封装度、可靠性和扩展性等方面的指标。关键需求决定架构设计，只有充分理解这些关键需求，才能更好地理解主流分布性消息队列架构设计的优缺点。

队列模型

从模型上，分布式消息队列主要分成两大类：点对点模型和发布/订阅模型。点对点模型是最为传统和常见的消息队列模型。从发送者和接收者的角度看，点对点队列模型支持一对一、一对多、多对多、多对一等多种配置方式。该模型主要具有以下一些特点。

- 一条消息只能由一个接收者接收一次。消息发送者将产生的消息放入队列中，接收者从消息队列中取走消息。消息一旦被一个接收者取走，该消息就从消息队列中被删除。
- 发送者和接收者之间不存在时间上的相关性：发送者发送一条消息后，即使接收者当时没有运行，再次运行后都能取出该消息。
- 可在运行时动态添加和删除发送者和接收者，即根据需要扩展或收缩分布式消息系统规模。
- 消息在队列中的放置顺序与发送顺序相同，但它们的使用顺序则取决于消息失效期、消息优先级等其他因素。

发布/订阅模型使得发送者和接收者之间的耦合关系变得更为松散，它们通过主题完成信息交换。发布/订阅模型与点对点模型最大的不同在于，发送者（即发布者）发送的每一条消息都可以被消息队列的所有接收者（即订阅者）接收一次，即每个接收者都有一次消费该消息的机会。发布/订阅模型主要具有以下一些特点。

- 多个发送者可向一个主题发布消息。
- 多个接收者可使用一个主题中的消息。
- 长期接收者可能处于活动状态，也可能处于非活动状态。在它们处于非活动状态时，消息队列需要为它们保留消息，等它们再次处于活动状态时再将消息推送给它们。
- 发送者和接收者之间存在时间上的相关性：接收者只能使用在它创建订阅后被发布的消息。

从架构设计的角度看，点对点模型实际上是发布/订阅模型及其他复杂模型的基础。在点对点模型实现的前提下，系统集中管理发布/订阅关系，根据订阅关系为每个订阅者单独创建一个点对点类的消息队列供其接收消息。在发布者发布一个消息后，系统根据订阅关系把消息复制到对应订阅者的接收队列供订阅者接收，即实现发布/订阅模型的支持。为简化分布式消息队列的架构设计，使之更专注于核心功能，目前很多轻量级消息队列往往仅提供对基本点对点队列模型的支持，应用要根据实际需要决定是否支持其他复杂模型。

高性能、高并发

性能是分布式消息队列的核心关注点，主要从两个方面进行度量，一方面是消息从发送到立即接收端到端全流程的时延，另一方面是指同时发送或者同时接收消息时系统并发吞吐。

可扩展性体现为：系统扩展后能够提供更快的访问速度、更大的存储空间；系统扩展必须平滑和线性，不能采用推倒重来的方式；系统扩展不影响系统正常提供服务。

可扩展性

在Web 2.0时代，随着云计算和移动互联网的兴起，诸如微博、网盘、应用商店和SNS社区等应用得到了快速的发展。这些面向公众服务的业务应用，对消息队列处理能力的要求非常高，往往不是几台服务器节点就能满足的。而且随着业务量日益增加，也要求消息队列能够随之扩展以重新满足业务量需求。可扩展性体现在以下几方面。

- 系统扩展后能够提供更快的访问速度和更大的存储空间，以满足应用大规模发展的需要。
- 系统扩展必须平滑和线性，不能采用推倒重来的方式，需要保留原有的数据，不改变原有系统的部署方式，同时将新增节点加入系统的操作必须简单易行。
- 系统扩展在线进行，不影响系统正常提供服务，扩展对应用完全透明。

可靠性

可靠性指消息在发送和接收的整个生命期中不会因为意外而丢失（如断网、重启、损坏等）。有些消息队列系统中的消息队列依附于连接。当连接断开后，该消息队列中的消息即随之销毁而造成

消息丢失。另外，有些消息队列不提供消息的持久化保存能力，消息只存储在内存中。当某个存储消息的服务器重启时，也会造成数据丢失。为实现消息的可靠存储，消息队列需要将消息独立于连接之外，同时将消息持久化至硬盘等持久介质中。为进一步提升可靠性级别，可以考虑将消息进行多副本存储，且副本分布于不同的物理位置，实现高等级的容灾。

多系统跨平台支持

能够提供多平台系统支持的分布式消息队列往往具备更广泛的使用空间。分布式消息队列通过对多平台系统的支持，屏蔽操作系统、网络协议、开发语言等方面的差异，方便企业应用实现异构系统的集成。比如，IBM WebSphere MQ可支持多种网络协议，提供多达35种操作系统的单一API支持，几乎可以连接任何系统，在企业应用领域广泛被使用。

在明确了分布式消息队列的关键需求和核心关注点后，接下来介绍下我们研发的基于P2P网络的分布式消息队列。

基于P2P网络的分布式消息队列

我们研发的分布式消息队列属于典型的轻量级分布式消息队列，支持JMS点对点队列模型。它部署非常简单，采用C/S架构，客户端以多种语言API方式提供应用调用。API接口高度封装，对应用开发屏蔽网络拓扑和通信协议。服务端基于P2P网络，消息队列本身及消息内容都分布存储在服务端P2P网络中各节点上，通过一致性HASH算法和虚节点机制相结合实现负荷均匀分布，消除数据访问热点。服务端处理能力线性扩展，扩展过程对应用完全透明。

基于P2P网络的分布式消息队列服务端采用分层架构设计法，从架构上分成分布存储层及逻辑控制层两个层次。

架构

P2P分布式消息队列架构，如图1所示。

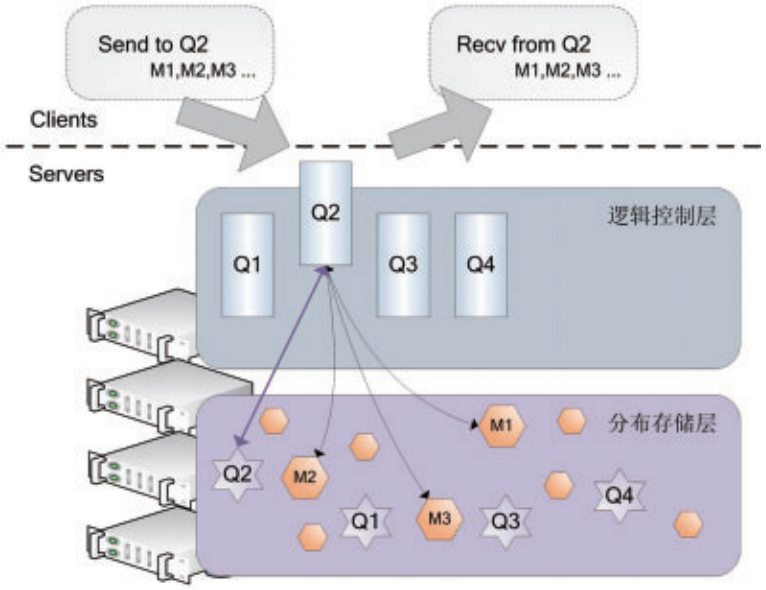


图1　P2P分布式消息队列架构

分布存储层基于P2P网络构建分布式K/V存储引擎，主要功能就是为逻辑控制层提供稳定可靠的数据存储。数据在分布式存储层的P2P网络中均匀分布，处理负荷在各节点趋于均衡；存储层可以根据逻辑控制层的需要设置不同的数据副本数，满足不同级别的数据可靠性需求；存储层根据应用具体需要提供多可靠性级别的存储介质管理机制，并采用高性能内存管理算法，减少碎片，提升内存管理效率。

逻辑控制层专注于分布式消息队列逻辑实现，提供针对分布式消息队列的创建、删除等管理接口、作为发送和接收者的客户端注册管理以及队列正常的消息收发功能。

在逻辑控制层，每个分布式消息队列被抽象成包括队列头指针、队列尾指针以及队列属性的独立对象。头指针指向消息队列中下一个新加入消息的位置，尾指针指向消息队列中未被取走的第一个消息，通过头尾指针的移动实现消息内容出入队列的操作。队列属性是在消息队列创建时由应用指定的，包括消息队列支持的最大消息个数以及默认消息有效期等。逻辑控制层调用存储层的K/V接口将这些消息队列对象和消息内容分布存储到P2P网络中，如图2所示。

P2P分布式消息队列对应用提供队列的创建/删除、注册/注销、消息的发送、主动和被动消息接

收等操作，易于使用（如图3所示）。

特性和优势

表1中描述了P2P分布式消息队列的特性和优势。

关键技术点

■ 基于一致性HASH和虚节点的分布算法

它是P2P网络的关键算法，通过一致性HASH和虚节点相结合，有效消除集群访问热点。

■ 智能路由交换技术

通过路由智能交换算法，分布式消息队列服务端P2P网络集群的路由变更能在服务器节点间、服务器和客户端API之间快速生效。分布式消息队列客户端API在本地也能实现路由信息的缓存，在数据访问的同时完成和服务节点的路由交换，大大提升了路由查找的效率，降低了数据访问的时延。

■ 分布式计数器

分布式计数器用于消息队列中对象的头指针和尾指针的递增控制。多个客户端可能同时对一个消息队列进行数据访问，通过分布式计数器机制实现头尾指针并发单向递增。

■ 灵活高效的数据存储引擎

分布式消息队列存储层支持基于内存和持久化模式的消息存储，满足不同应用的需求。存储层通过接口标准化实现持久化引擎可插拔，目前支持两种持久化存储引擎。

BDB（BerkeleyDB）：以嵌入式K/V数据库方式提供海量数据存储。

IFS（Indexed Fast Storage）：基于内存索引的顺序持久化模式存储。IFS模式的目标是尽量将磁盘的随机写变成顺序写，最大限度地提高磁盘读写的I/O吞吐。

■ 可靠、及时的消息到达通知机制

当应用通过注册回调函数的方式被动接收消息时，分布式消息队列通过两种手段确保消息到达时能够及时通知应用获取。服务端统一管理每个分布式消息队列对象当前激活客户端。当消息队

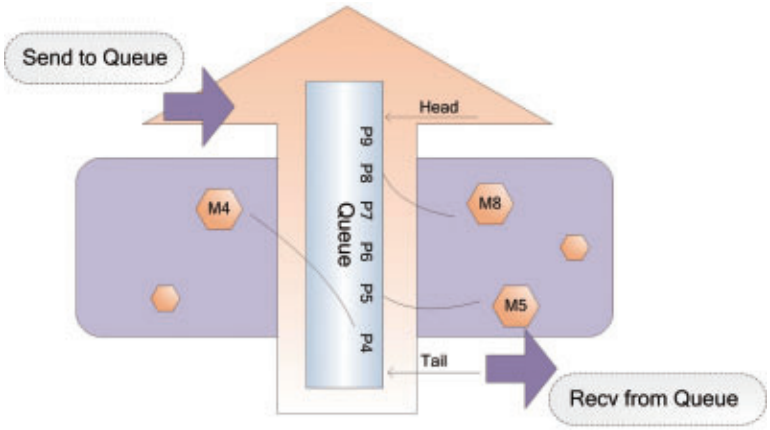


图2　P2P消息队列逻辑操作



图3　P2P消息队列协作场景

表1　P2P分布式消息队列的特性

特性	描述
高性能	性能对比测试大幅领先于同类产品，同时通过服务器集群扩展实现处理吞吐线性扩充。比如在同等硬件和网络条件下进行1000个消息队列的收发测试，RabbitMQ速度是每秒3万多，P2P消息队列达到接近5万
扩展更灵活实现海量存储	分布式消息队列对象和消息队列存储的消息内容都通过K/V形式存储在P2P网络中，因此通过P2P网络的扩展，一方面能够支持海量的分布式消息队列，另一方面消息队列存储的内容也能够无限扩展
负荷均衡，热点消除	通过一致性HASH和虚节点策略，针对不同消息队列和同一消息队列中不同内容的访问将路由到P2P集群中不同的节点上，从而达到负荷处理均衡的效果，通过对虚节点路由的调整能有效消除P2P集群访问的热点
数据可靠	P2P消息队列可靠性主要依靠分布式环境中数据多副本来保证，副本策略是根据应用需要可配置的，并且能够实现不同机房间的异地数据容灾。支持基于包括SSD、硬盘等持久化介质的数据存储。考虑到硬盘容量及I/O性能限制，持久化存储通常采用专用定制化大容量存储服务器实现。此外，还能通过一系列数据修复策略，在网络或硬件设备发生故障并排除后，将丢失的数据进行恢复并维持副本间的数据一致性
高度封装、管理简单	消息队列客户端API高度封装，简单易用。同时提供系统拓扑结构调整、负载均衡、性能、事件和警报监控、DASHBOARD等监控机制和管理手段

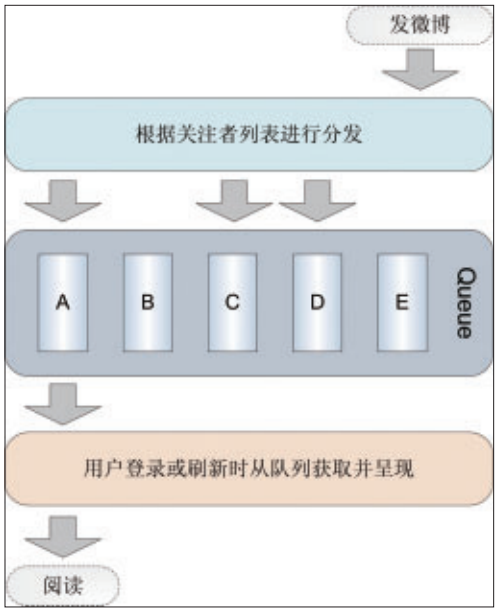


图4　P2P消息队列在微博中的应用

列有新消息达到时，服务器会根据一定的算法从该消息队列激活的客户端连接中选一条发送消息到达通知，客户端收到到达通知后执行应用注册的回调函数通知应用。这种通知方式具有较高实时性，有效降低了消息从进入队列到被应用取走的端到端时延。同时，为防止服务端主动的消息到达通知由于网络抖动等原因丢失，客户端API内部也会定期主动和服务端通信轮询是否有新消息到达，轮询过程对应用透明。

应用场景

通过P2P分布式消息队列可以把效率较低的同步应用变成高效的异步应用，把复杂的紧耦合系统变为简单的松耦合系统，实现不同系统和不同应用间高效的消息交互，同时也使大规模应用的消息广播和订阅功能平台化成为可能，使应用开发更容易。目前该产品已经在即时消息IM、企业协同、企业消息总线（ESB）、微博、云存储等应用中得到了广泛的应用。图4是P2P分布式消息队列在微博产品中的应用场景。

微博是分布式消息队列的一个典型使用场景，每天会根据用户的行为产生海量动作流（Activity Stream）数据，在这里引入一个高吞吐的P2P分布式消息队列是非常必要的。

在微博应用中，每个微博用户都有一个粉丝列表和关注列表，同时每个用户都拥有一个属于自己的分布式消息队列。当用户发布一条新微博时，后台程序会根据其粉丝列表，将微博消息异步地群发至其粉丝们的消息队列中。一段时间后，每个用户的消息队列里都会有大量由其他用户（关注列表）发来的微博消息。在用户登录或刷新页面时，后台会从该用户的消息队列中获取来自关注列表的微博消息，并提交至前台进行规整和呈现，供微博用户阅读。

通过引入P2P分布式消息队列，不仅能够满足微博这种社区类应用的高性能、高并发的需求，而且易于根据用户量的不断攀升对现有集群进行线性扩容，此外在单点故障容错性等方面也有不俗的表现。

总结

P2P分布式消息队列凭借负荷均衡、处理能力线性扩展、海量消息存储等优势，不仅适用于企业级应用，更适合在移动互联网、云计算等面向公众服务的大规模应用领域作为基础架构实现高性能消息处理。P2P分布式消息队列所解决的问题都是云计算架构中的关键问题，应用能够借助其高性能、高可靠、可扩展的特性以及优秀的分布式架构解决难题，提升自身的产品价值。



高洪

中兴通讯业务研究院平台研发总工，先后从事智能网和增值业务的研发设计工作，研究方向为IN技术、信令网络、开放业务平台，IMS业务。目前主要从事云计算平台研发工作。

责任编辑：杨爽（yangshuang@csdn.net）

个性化推荐技术的十大挑战

文 / 周涛

个性化推荐经常被人误解为细分市场和精准营销这两个概念。虽然它们之间有一些联系，但实质上却相差甚远。本文不仅清楚地讲述了个性化推荐技术，更列出了其所面临的十大挑战。

很多人都知道个性化推荐，却有不少认识上的误区。有的人认为个性化推荐就是细分市场和精准营销，但实际上细分市场和精准营销往往是把潜在的用户分成很多群体，这与基于全体的统计相比固然有了长足的进步，但距离“为每一个用户量身定做的信息服务”还有很大的差距，只有做个性化才能实现亚马逊CEO Jeff Bezos的梦想“如果我有100万个用户，我就为他们做100万个亚马逊网站”。所以，市场细分只是初级阶段，而个性化推荐则是市场细分的极致。还有人认为个性化推荐等同于协同过滤，这可能是因为协同过滤出现得比较早并且比较容易被大众理解，但实际上协同过滤只不过是众多个性化推荐技术中的一个早期成员，远不能代表个性化推荐技术。

随着个性化的商业应用延展到用户生活信息流的方方面面，个性化推荐技术也在日新月异不断发展，类似于协同过滤这样的早期技术已经不能满足新环境下的要求，无法解决新出现的问题。令人兴奋的是，过去十年中，我们见证了无数顶尖级专家与学者投身于推荐方法与技术的创新中。今天，还些人认为个性化推荐技术的研究已经进入了很成熟阶段，没有什么特别激动人心的问题和成果。事实却恰恰相反，现在个性化推荐技术面临很大的挑战，我们之前只看到了个性化推荐技术难题这座冰山所露出的一角。

本文将列出个性化推荐技术所面临的十个挑战（仅代表个人观点），其中有一些是很多年前就认识到但一直没有得到解决的长期问题。事实上，有些挑战是不可能完全解决的，只能提出改良方案，还有一些是在最近研究中提出来的焦点问题。

1 数据稀疏性问题

现在推荐系统规模越来越大，用户和商品（包括音乐、网页、文献等物品）数目动辄百千万计，而且用户之间选择的重叠非常少。如果以用户和商品之间已有的选择关系占所有可能存在的选择关系的比例来衡量系统的稀疏性，那么我们平时研究最多的MovieLens数据集的稀疏度是4.5%，Netflix是1.2%，Bibsonomy是0.35%，Delicious是0.046%，这些其实是非常密的数据。

想想淘宝上号称有8亿商品，平均而言一个用户能浏览800件吗？估计不能，因此稀疏度应该在百万分之一或以下的量级。数据非常稀疏，使得绝大部分基于关联分析的算法（如协同过滤）效果都不好。本质上，这个问题是无法完全克服的。解决这个问题的办法很多，例如可以通过扩散的算法，从原来的一阶关联（两个用户有多少相似打分或者共同购买的商品）到二阶甚至更高阶的关联（假设关联性或者说相似性本身是可以传播的），也可以添加一些默认的打分，从而提高相似性的分辨率。一般而言，数据规模越大就

越稀疏。现在认为能够处理稀疏数据的算法（如扩散、迭代寻优、转移相似性等）更有价值。

2 冷启动问题

因为新用户罕有可以利用的行为信息，所以很难给出精确的推荐。反过来，由于新商品被选择次数很少，也难以找到合适的办法推荐给用户。一种办法是利用文本信息进行辅助推荐，或者通过注册和询问得知一些用户的属性信息，比如年龄、居住城市、受教育程度、性别和职业等。

最近广泛应用的标签系统提供了解决冷启动问题的可能方案，因为标签既可以看作是商品内容的萃取，同时也反映了用户的个性化喜好。以《桃姐》这部电影为例，有的人打的标签是“伦理”，有的人打的标签是“刘德华”，两个人看的电影一样，但兴趣点可能不尽相同。当然，利用标签也只能是提高有少量行为的用户的推荐准确性，对于纯粹的冷启动用户，没有任何帮助，因为这些人还没有打过任何标签。

有趣的是，最近的研究显示，新用户更容易选择特别流行的商品。无论如何，这都是一个好消息，说明使用热销榜也能获得不错的结果。冷启动问题还可以通过多维数据的交叉推荐部分解决，其精确度和多样性又远胜于热销榜，这一点后面会进一步介绍。

3 大数据处理与增量计算问题

因为数据很稀疏，大部分数据都拥有百千万计的用户和商品，所以如何快速高效地处理这些数据成为迫在眉睫的问题。而算法时间和空间上的复杂性，尤其是前者，得到了空前重视。一个高效的算法，要么复杂性很低，要么能够很好地并行化，要么两者兼具。局部扩散算法在这两个方面都具有明显优势。

另一种解决方法是设计增量算法。也就是说，当产生新用户、新商品和新的连接关系时，算法的结果不需要在整个数据集上重新进行计算，而只需考虑所增加节点和连边局部的信息，对原有的结果进行微扰，快速得到新结果。一般而言，随着加入的信息量的增多，这种算法的误差会积累变大，最终每过一段时间还是需要利用全局数据重新进行计算。

一个特别困难的挑战是如何设计一种能够保证其误差不会累积的算法，也就是说其结果与利用全部数据重新计算的结果之间的差异不会单调上升。我们把这种算法叫做自适应算法，它是增量算法的一个加强版本，其设计要求和难度更高。

现在业界已经在应用增量算法。例如，百分点科技推荐引擎中的若干算法都采用了增量技术，使得用户每次浏览收藏或者购买商品后其推荐列表立刻得到更新。当然，只是该引擎的部分算法实现了增量技术，没有达到所有算法都能够自适应学习的程度，还有很长的路要走。

4 多样性与精确性的两难困境

如果要给用户推荐他喜欢的商品，最“保险”的方式就是给他特别流行或者得分特别高的商品，因为这些商品有更可能被喜欢（至少Bezos会这么想），往坏了说，也很难特别被讨厌。但这种推荐产生的用户体验并不一定好，因为用户很可能已经知道这些热销或流行的产品，所以得到的信息量很少，并且用户不会认为这是一种“个性化”推荐。

事实上，Mcnee等人已经警告大家，盲目崇拜精确性指标可能会伤害推荐系统，因为这样可能会导致用户得到一些信息量为0的“精准推荐”并且视野变得越来越狭窄。让用户视野变得狭窄是协同过滤算法的一个主要缺陷。与此同时，应用个性化推荐技术的商家，也希望推荐中有更多的品类出现，从而激发用户新的购物需求。

遗憾的是，推荐多样的商品和新颖的商品与推荐的精确性之间存在矛盾，因为前者风险很大——推荐一个没人看过或者打分较低的东西，很可能被用户憎恶，从而效果更差。很多时候，这是一个两难的问题，只能通过牺牲多样性来提高精确性，或者牺牲精确性来提高多样性。一种可行之策是直接对推荐列表进行处理，从而提升其多样性。这种方法固然在应用上是有效的，但没有任何理论的基础和优美性可言，只能算一种实用的

招数。

我们发现，精巧混合精确性高和多样性好的两种算法，可以同时提高算法的多样性和精确性，不需要牺牲任何一方。遗憾的是，我们还没有办法就这个结果提供清晰的解读和深刻的见解。多样性和精确性之间错综复杂的关系和隐匿其后的竞争，到目前为止还是一个很棘手的难题。

5 推荐系统的脆弱性.问题

推荐系统在电子商务领域能带来重大的经济利益的驱动，导致一些心怀不轨的用户会提供一些虚假或恶意的行为，故意增加或者压制某些商品被推荐的可能性。因此，一个算法能否在一定程度上保持对恶意攻击的鲁棒性，成为需要认真考虑的一个特征。以最简单的关联规则挖掘算法为例，Apriori算法的鲁棒性就远胜于k近邻算法。

已有一些技术专门用来提高推荐系统面对恶意攻击的鲁棒性。比如，通过分析对比真实用户和疑似恶意用户之间打分行为模式的差异，提前对恶意行为进行判断，从而阻止其进入系统或将恶意用户的影响力降至最低。

总体来说，这方面的研究相对较少，系统性的分析还很缺乏，反而是攻击策略层出不穷，有种“道高一尺，魔高一丈”的感觉。Burke等人2011年的研究报告“Robust Collaborative Recommendation”中就分析了4大种类、8种不同的攻击策略。

6 用户行为模式的挖掘和利用

深入挖掘用户的行为模式有望提高推荐的效果或在更复杂的场景下进行推荐。比如，新用户和老用户具有很不一样的选择模式：通常，新用户倾向于选择热门商品，而老用户对小众商品关注更多；新用户所选择的商品相似度更高，老用户所选择的商品多样性较高。

有些混合算法可以通过一个参数调节推荐结果的多样性和热门程度。在这种情况下，就可以考虑为不同用户赋予不同参数（从算法结果的个性化到算法本身的个性化），甚至允许用户自己移动一个滑钮调节这个参数——当用户想看热门产品时，算法提供热门推荐；当用户想找点很酷的产品时，算法也可以提供冷门推荐。用户行为的时空统计特性也可以用于提高推荐或者设计针对特定场景的应用。

用户的选择可能同时蕴含了长期的兴趣和短期的兴趣，通过将这两种效应分离出来，可以明显提高推荐的精确度。事实上，假设用户兴趣随时间按照指数递减，也能够得到改进的推荐效果。

深入挖掘用户的行为模式有望提高推荐的效果或在更复杂的场景下进行推荐。有些混淆算法能通过一个参数调节推荐结果的多样性和复杂程度。

现在用手机上网已经越来越普及，与此同时，嵌入GPS的手机越来越多，因此，基于位置的服务成为一个受到学术界和业界广泛关注的问题。基于位置信息的推荐可能会成为个性化推荐的一个研究热点和重要的应用场景，而解决这个问题需要能够对用户的移动模式有深入理解（包括预测用户的移动轨迹和判断用户在当前位置是否有可能进行餐饮购物活动等），同时还要有定量的办法定义用户之间以及地点之间的相似性。另外，不同用户打分的模式也很不一样，用户针对不同商品的行为模式也不一样（想象一下在网上下载一首歌和团购房子时的区别），这些都可以用来提高推荐的效果。

7 推荐系统效果评估

推荐系统的概念已经提出几十年了，但怎样评价推荐系统，仍然是一个很大的问题。常见的评估指标可以分为四大类，分别是准确度、多样性、新颖性和覆盖率。每一类还有不同的指标。比如，准确度指标还可分为四大类，分别是预测评分准确度、预测评分关联、分类准确度和排序准确度。以分类准确度为例，又包括准确率、召回率、准确率提高率、召回率提高率、F1指标和AUC值。

朱郁筱和吕琳媛撰写的《推荐系统评价综述》一

文几乎总结了文献中曾经出现过的所有推荐系统指标，这些指标都是基于数据本身的指标，可以认为是第一层次。实际上，在真实应用时，更为重要的是另外两个层次的评价。第二个层次是商业应用上的关键表现指标，如受推荐影响的转化率、购买率、客单价、购买品类数等。第三个层次是用户真实的体验。

绝大部分研究只针对第一个层次的评价指标，而业界真正感兴趣的是第二个层次的评价（比如，到底是哪个指标或者哪些指标组合的结果能够提高用户购买的客单价），而第三个层次最难，没人能知道，只能通过第二层次来估计。因此，如何建立第一层次和第二层次指标之间的关系，就成为了关键。这一步打通了，理论和应用之间的屏障就通一大半了。

8 用户界面与用户体验

与其说，这是一个学术性质的问题，不如说是真实应用的问题。十年前就有学者指出，推荐结果的可解释性，对于用户体验有至关重要的影响——用户希望知道这个推荐是怎么来的。在这个意义上，协同过滤有明显的优势。

基于商品的协同过滤，亚马逊在发送推荐的电子邮件时会告诉用户之所以向其推荐某书，是因为用户以前购买过某些书。相对地，矩阵分解或者集成学习算法就很难向用户解释推荐结果的起源。用户更喜欢来自朋友的推荐而不是系统的推荐，这一点在后面还会详细提到。

另外，推荐列表往往含有很多项，这些推荐项最好能够分成很多类别，不同类别往往来自于不同的推荐方法。比如，看过还看过（浏览过本商品的客户还浏览过的商品）、买过还买过（购买过本商品的客户还购买过的商品）、看过最终购买（浏览过本商品的客户最终购买的商品）、个性化热销榜（个性化流行品推荐）和猜你喜欢（个性化冷门商品推荐）等。

当然，如何更好呈现推荐，是一个很难建立理论模型和进行量化的问题。对于不同的被推荐品而言，用户界面设计的准则也可能大不相同。比如，在首页、类目页、具体商品页、购物车页都应该分别放置哪些推荐栏？不同的推荐栏放在页面的什么位置能够优化用户体验？什么情况下应该让用户自己选择推荐的个性化程度？基于用户行为的实证研究可以回答一部分问题，同时需要建立一个可以进行A/B测试的系统，或可积累重要的实验数据。

9 多维数据的交叉利用

目前网络科学研究一个广受关注的概念是具有相互作用的网络的结构和动力学。网络与网络之间的相互作用大体可以分成三类。第一类是依存关系。比如，电力网络和Internet，如果发生了大规模停电事故，当地的自主系统和路由器也会受到影响，导致网络局部中断。第二类是合作关系，比如人的一次出行，可以看作航空网络、铁路网络和公路网络的一次合作。第三类是交叠关系，主要针对社会网络，这也是我们最关注的。

几乎我们每一个人都参与了不止一个大型的社会网络，既有新浪微博的账号，又是人人网的注册用户，还是手机用户，那么你已经同时在三个巨大的社会网络中了。与此同时，你可能还经常在淘宝、京东、麦包包、1号店、库巴网等网站上进行网购，那么你就成为了一张巨大的用户–商品图中的一员。

想象如果能够把这些网络数据整合起来，特别是知道每个节点身份的对应关系（不需要知道真实身份，只需要知道不同网络中存在的一些节点是同一个人），那么其中定会有特别巨大的社会经济价值。举个例子，你可能已经在新浪微博上关注了很多数据挖掘达人的微博，并且分享了很多算法学习的心得和问题。而当你第一次上当当网购书时，主页向你推荐数据挖掘的最新专著并附有折扣，你会心动吗？交叠社会关系中的数据挖掘，或称多维数据挖掘，是真正解决系统内部冷启动问题的终极法宝——只要用户在系统外部的其他系统有过活动。

单纯从个性化商品推荐来讲，可以利用用户在其他电商网站的浏览购买历史来提高在目标电商推荐的精确度——当然，每一个电商既是付出者，

也是获利者。总体而言，大家能够通过提高用户体验和点击深度实现共赢。与此同时，可以利用微博和其他社会网络的活动提高商品推荐的精度，还可以反过来利用商品浏览历史提高微博关注对象推荐的精度。推荐一个经常购买专业羽毛球和浏览各种专业羽毛球设备的用户关注羽毛球的专业选手和业余教练的成功率应该很高，而且不会陷入“总在一个圈子里面来回推荐”的毛病中。从机器学习的角度，杨强等人提出的“迁移学习”算法有望用来解决这种跨邻域的推荐。

我们分析了百分点科技服务客户的真实数据，发现有相当比例的用户都具有交叉购物的习惯（在多个独立B2C电商有浏览和购买行为）。即便只考虑两个点，例如利用麦包包的浏览购买数据为拍鞋网的用户进行个性化推荐（这些用户是在拍鞋网上没有任何历史记录的新用户，但在麦包包上有浏览购买行为），就可以明显提高推荐的准确度（比完全冷启动的随机推荐高数十倍）。而如果利用3家或以上的外部电商的数据，其推荐的精确度可以明显高于热销榜（注意，热销榜一点个性化都没有），特别在团购类网站上表现非常好。

虽然针对多维数据挖掘的研究刚起步，但我们完全可以相信这在学术和应用上都将是一个焦点和难点。

10 社会推荐

很早以前，研究人员就发现，用户更喜欢来自朋友的推荐而不是被系统“算出来的推荐”。社会影响力被认为比历史行为的相似性更加重要。例如通过社会关系的分析，可以大幅度提高从科研文献到网购商品推荐的精确度。来自朋友的社会推荐有两方面效果：一是增加销售（含下载、阅读……），二是在销售后提高用户的评价。

社会推荐的效果也不完全是正面的，譬如Leskovec等人在论文“The Dynamics of Viral Marketing”中举了一个反例：朋友推荐对书的销售增长几乎没有帮助，有时会起到负面作用。

在社会推荐方面，国内做得最出色的是豆瓣网，其朋友推荐被接受被高度评价的比例非常高。最近有证据显示，朋友推荐也是淘宝商品销售一个非常重要的驱动力量。

社会推荐方向存在的挑战主要可以分为三类：一是如何利用社会关系提高推荐的精确度；二是如何建立更好的机制以促进社会推荐；三是如何将社会信任关系引入到推荐系统中。社会推荐的效果可能来自于类似口碑传播的社会影响力，也可能是因为朋友之间本来就具有相似的兴趣或者兴趣相投更易成为朋友。对这些不同的潜在因素进行量化区别，也是学术研究的热点之一。

相关阅读

有些朋友可能还不是很了解个性化推荐，推荐阅读《自然科学进展》上的综述，能快速得到很多信息，了解个性化推荐研究的概貌。有了这个基础，如果想要了解应用级别的算法和技术，推荐阅读项亮、陈义和王益合著的《推荐系统实践》一书。

同时建议关注国外最近出版的《Recommender Systems Handbook: A Complete Guide for Scientists and Practioners》和《Recommender Systems: An Introduction.》。前者实际上是很多文章的汇总，写这些文章的作者都是技术专家，文章质量不错。Adomavicius和Tuzhilin撰写的“Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions”一文特别有影响力，不仅系统回顾了推荐系统研究的全貌，还提出了一些有趣的开放性问题。今年在《物理报导》上有一篇关于推荐系统的大综述，应该是目前最全面的，所强调的不仅是算法，还有很多现象和思路，推荐大家阅读。



周涛

电子科技大学互联网科学中心主任、教授、博士生导师，阿里巴巴商学院钱塘特聘教授，北京计算科学研究中心客座教授。毕业于瑞士弗里堡大学，获物理系哲学博士。

责任编辑：杨爽（yangshuang@csdn.net）

ThoughtWorks技术雷达（2012年3月）

文 / ThoughtWorks技术战略委员会　译 / 唱鑫，张晓蕴，史磊

自2011年7月《技术雷达》发布后，以下几个趋势越来越明显：SQL数据存储替代产品的持续发展，谨慎对待从UI到测试的所有代码，基于浏览器的语言和技术更加严谨与多样性化，应用程序和服务更小、更简单、更快速。

《技术雷达》是将ThoughtWorks资深技术专家的讨论结果，用图解的形式，将所有项目分为技术、工具、语言和平台几大类，为开发人员提供的技术趋势指南。其中新项目或有新动向的用三角代表，之前已经提到的项目用圆形代表。下文将各项内容定义为四个阶段。

- 保留（Hold）：应当谨慎使用的技术。
- 评估（Assess）：值得探索的技术，以了解其对公司的影响。
- 试用（Trial）：值得应用的技术。公司有必要了解如何使用，并且已经可以在风险可控的项目中试用。
- 应用（Adopt）：强烈建议公司使用的技术。公司应在适当的时候应用于项目上。

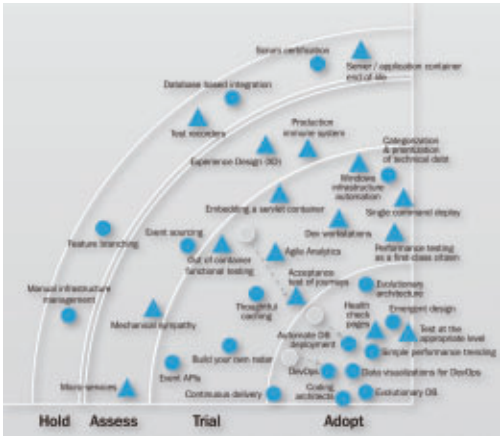


图1　技术趋势雷达

技术

浮现式设计是敏捷工程中的一种高级实践，目前对该领域的研究十分活跃。系统架构应由系统中相关技术的需求驱动产生，而不应基于对未来可能产生变化的主观推断确定。我们已经至少识别出两种浮现式设计方法：基于精益软件原则的最后责任时刻（Last Responsible Moment）适用于新建项目，而积累惯用模式（Idiomatic Patterns）则更适用于已有项目。

微服务（Micro-services）部署时通常不依赖容器，或者使用内嵌的HTTP服务器，这种方式摆脱了传统的大型技术服务模式。通过增加运营和维护期的复杂度换取了更高的可维护性。而运营和维护期的复杂度可以通过采用基础设施自动化和持续部署的相关技术加以降低。总的来说，微服务是管理技术债和应对扩展差异性的有效方式，尤其是部署在围绕业务能力构建的面向服务架构中。

十多年前的内存十分昂贵，作为在共享服务器或集群上运行和管理多个应用程序的手段，应用程序服务器技术非常有价值，因此深受欢迎。如

今，应用程序更多地运行于独立的物理或虚拟服务器上，对应用程序服务器的需求大幅减少。我们建议在你的组织中评估是否停止使用应用服务器，而只有在这样复杂的技术能给你带来额外收益时才使用它。

在Java应用中嵌入诸如Jetty这样的嵌入式servlet容器，要比让应用运行在一个容器中有更多好处。由于应用的启动过程更简单、开发环境更接近生产环境，因此测试变得相对容易。因为多个应用无需共享库或驱动程序，所以不会出现库或驱动版本不匹配等令人讨厌的问题。虽然采用这种模型需要在生产环境下同时管理和监控多个Java虚拟机，但这种模型的简单性和隔离性所带来的优势更为显著。

随着嵌入式HTTP服务器越来越受欢迎，脱离容器的功能测试也越来越普及。通过采用模拟容器，你可以在系统的边界上构建测试，这样既可以实现快速反馈，又能获得较高的测试覆盖率。诸如Jetty等服务器和.NET平台上的Plasma工具可以大大降低运行测试集所需的时间。

目前有一个令人担忧的趋势，那就是开发人员与运行代码的硬件越来越远。开发和维护之间日益增长的虚拟化和分离使这种情况变得更糟。与之形成鲜明对比的是某些团队正在编写可以获得机械同情（Mechanical Sympathy）的代码，这种代码可以使软件获得难以想象的性能提高。例如Java平台上的LMAX Disruptor开源项目就是一个例子。对于金融和大数据等对性能有很高要求的领域，越贴近底层就越能获得更高的回报。

将敏捷方法应用于数据仓库、商业智能和敏捷分析，可以获得更好的回报，并且可以提高商业响应速度。与传统的批量更新相比，通过采用轻量级的技术，如REST服务等来实现，几乎可以达到实时地更新数据。这样可以从应用中获取用户的行为数据，以支持更好的用户体验和数据可视化效果。

我们发现在应用中添加简单的健康检查页面非常有用。这样人们可以快速了解单个节点的健康状况。我们经常对其进行扩展并添加如订单号、错误率或类似的度量信息。使用简单的嵌入式Web服务器，甚至可以让非Web的服务也很容易地通过HTTP暴露内部的信息。通过使用微格式，这些网页可以很容易地被其他监控工具所使用，从而成为全面监控系统的一部分。

用户故事级别的测试会导致将重点放在完成单个故事，而不是耦合的功能上。这样往往会导致开发出一组过于庞大、运行缓慢且难以维护的测试套件，从而导致反馈时间的延长。一个可选方案是使用用户旅程（User Journey），即将用户故事分组然后抽象出用户交互行为，从而为用户和业务双方提供价值。将用户旅程作为自动化的测试套件，可以在相对长的时间内和用户需求匹配，这类测试如果失败，则表明应用程序不能满足用户的需求。

一些行为驱动设计的测试框架（如Cucumber）与浏览器端自动化测试框架（比如Selenium）的组合，使得浏览器级别的验收测试被广泛使用。这样做的一个缺点就是增加了大量运行成本很高的测试。相反，我们应该在合适的粒度去进行测试，尽可能地贴近代码本身，这样才能最大程度提高测试运行的效率。浏览器级别的测试应该处于金字塔塔尖，下面应该是相应层的验收测试与单元测试。

测试录制工具看似很有价值，因为它们提供了一种快速捕获录制用户对应用程序操作的方式。然而我们反对将测试录制工具作为一种常用工具使用，因为录制的测试往往非常脆弱，会因为UI的轻微变化而导致失败。测试录制工具产生的测试代码质量往往相对较差，经常出现不必要的重复。更重要的是，测试录制工具会阻碍测试自动化团队和开发团队之间的沟通。当面对一个很难通过用户界面进行测试的应用时，团队之间进行一次重要的谈话从而构建一个更容易测试的UI。

工具

混合持久化是指，根据效率和数据使用方式的差异而使用不同数据存储方式的技术。不是所有的应用场景都适合使用RDBMS作为默认的数据库。相反，先问这样的问题：session相关数据是否应该保存在这个数据库中？还是应该保存在某

个Key-Value存储系统中？客户和产品的关系数据是否应该保存于图形数据库中？即使是在一个应用程序内，使用如MongoDB、Riak或Neo4J等NoSQL数据库，也会让我们重新考虑如何管理。

Riak是一个无模式、无数据类型的分布式Key-Value存储系统。它在写入诸如session、购物车、日志流等大数据时很有优势。它的分布式簇的自我恢复、簇间分布数据的一致性保证、冲突检测并自动修复等功能，在需要高可用性的环境下会非常有用，并且可以为架构设计提供有趣的解决方案。

随着移动应用的兴起，JavaScript的文件大小和性能变得尤为重要。作为对一些大库的直接回应，JavaScript微框架出现了。这些轻量级的框架只做一件事情，比如DOM操作或MVC，并且大小在1KB内。通过一些微框架的组合使用，开发人员就可以获得他们想要的所有功能，并且不需要大库的开销。Microjs.com里有大量这样的微框架，当然也有工具可以将这些微框架捆绑成一个单独的库。

JavaScript现在已经被认为是一种功能强大的主流语言，可以在多种客户端和服务器环境下使用。随着JavaScript代码库的不断扩展，需要有更多JavaScript工具的支持，尤其是在持续集成和测试领域。来自于不断成长且充满活力的社区的Backbone.js、SpineJS、JavaScriptMVC、Jasmine、JSTestDriver和HRcov等工具正在向前端发展。

Frank是一个开源库，可以使用Cucumber编写iPhone和iPad的功能测试，并在远程设备上执行。以前在iOS上，验收测试驱动的开发非常麻烦，该库填补了一项重要的空白。

虽然过去几年中MVC一直是Web开发的主流模式，但大多数库和框架未能遵守其最重要的原则之一：保证View层无逻辑。如果不能保证View层没有逻辑，会增加应用程序复杂性，使其难于测试，代码无法重用。最近出现的一些DSL，如Mustache开始扭转这种局面，它们适用于多种语言和平台，并且允许使用任意工具编辑，无需额外的语言支持，对UI开发和应用程序的总体设计很有帮助。



图2　工具趋势雷达

使用原生操作系统包部署组件及其依赖有许多优点，然而使用工具创建Linux原生包的过程却比较繁琐。FPM则是一个有用的工具，可以轻松创建RPM、DEB或Solaris包，减少了很多麻烦。

包管理系统是一种被广泛采用的实践方法，用于整合第三方类库。像RubyGems、Maven、APT等工具在语言和系统层面均可支持。NuGet是.NET平台上的包管理系统，它由一个Visual Studio IDE扩展和一个PowerShell模块组成，为进一步实现在.NET平台上的自动化构建提供了可能。

PSake是用PowerShell实现的自动化构建工具。它提供了简明的语法来声明构建任务和依赖，而不使用XML。并且还可以在构建脚本内使用PowerShell和.NET框架的所有功能。

平台

Node.js只是越来越受欢迎的异步I/O单线程服务器的一个例子。传统的Web或应用服务器为每个访问请求创建一个线程，当与该访问请求相关联的所有处理任务都完成且收到响应之后才释放资源。如果其中任意一个处理任务需要占用I/O，则在I/O发生时线程会被阻塞。这种方法导致每个连接都会占用一个线程，而不管该线程实际是否消耗CPU周期，所以会浪费如文件描述符和内

存等资源。现今开始出现一种新的服务器架构，它使用单线程来服务多个连接，可以异步处理I/O。这种架构已经出现在如下平台上：Node.js、Nginx，以及Webbit。这些服务器因为处理每个连接所占用的资源非常少，所以处理并发连接的能力比传统的Web服务器要高上很多量级。

出于对隐私和安全的担心，或是为了赋予已投资的硬件新用途，许多公司开始选择建立私有云。针对此市场有多种开源产品以及商业产品。但计算、存储和网络管理仅仅是应用私有云的起点。私有云中许多服务和流程必须要定制实施，这样才足以和外部Amazon、Rackspace或者其他开放云平台提供的公有云服务进行竞争。

混合云描述了集成公有云和私有数据中心最佳特性的一组模式。混合云允许应用程序在正常时间运行于私有数据中心，而高峰时段的过载请求则运行于公有云中的租赁空间。另一种集成公有云和私有云的敏捷的方法是利用公有云的灵活性和延展性来开发和了解应用的生产环境特性，然后在其稳定时移入私有数据中心的永久架构内。

虽然用户很容易忽略基于云服务的地理位置，但在考虑选择合适的平台时，因为法律或者技术方面的原因，地理位置反而是一个严重的制约因素。随着Amazon最近加入了巴西和新加坡区域，对于以前处于IaaS服务不太好的地区的人们来说，基于AWS的系统将变得更加切实可行。此外，Amazon持续地为已有服务添加新特性（比如VPC），因此在对资源初始化的灵活性要求很高的场景下，我们仍然推荐使用AWS。

我们发现许多企业都开始建立其自己的内部云部署环境，这些环境在搭建开发和测试环境时很容易被复制。很多情况下，创建环境的过程可以实现自动化，只需一个按键，开发人员就可以创建一组可以运行企业核心业务和协作系统的主机。从某种意义上讲，这是为内部客户提供的特定领域的PaaS。

Windows Phone 7令人惊讶地挺过了Windows平台长时间的危机。经过多次失败尝试后，微软不仅生产出用户体验与行业内其他主要竞争者相当的手机操作系统，而且还提供了配套的开发支

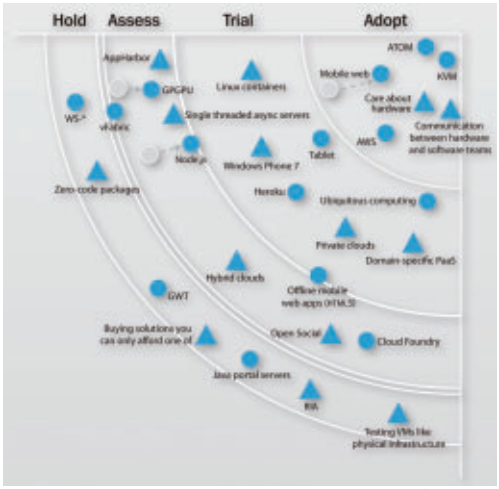


图3　平台趋势雷达

持。微软正在使Windows Phone 7成为一个强有力的竞争者，以及成为企业竞技场上整合体验更好的另一个选择。然而是否能够影响用户的选择还有待观察。

随着虚拟化不断发展，有些公司将虚拟机当成物理设备。我们不赞成为每个虚拟机都安装全套的操作系统或使用虚拟机进行负载测试。虚拟机可以被克隆、快照和操作，这些都是实际机器做不到的。虚拟机也有与实际机器硬件非常不同的性能特点。使用虚拟机时应全面了解其优点和局限性，否则你真的会遇到麻烦。

许多团队会遇到一些问题，这些问题是因为测试环境缺少一个只在生产环境中才具备的昂贵的硬件组件。而在许多情况下，预生产环境并不能达到生产环境的规模，因为不具备生产环境下的所有组件，模拟实际的情况。如果只买得起一个解决方案，如SAN、防火墙或负载平衡器，那我们建议不要购买这样的解决方案，因为这会导致只能在生产环境中进行实际测试，而其他环境都不具备这样的能力。

长期以来，我们一直对Flash和Silverlight等RIA技术不太热情，因为潜在的供应商封锁，对敏捷工程实践的支持不到位，以及潜在的过度使用等问题。现在看起来大厂商也开始同意我们了。因为较新版本的HTML可以处理大多数以前需要RIA才能解决的常见问题。我们认为新项目在使用任何这样的技术之前必须理由充分，并审慎地进行战略考虑。

目前市面上有大量的企业软件包，声称可采用零编码提供灵活的功能。这当然是一个有吸引力的概念——非技术的商业用户可以根据业务的独特要求配置软件，而不需要学习编程语言或雇用专业软件开发人员。但应当牢记，影响生产环境中软件行为的任何变化，无论是编码、配置、数据还是环境，都会导致业务系统缺陷或故障。编码只是专业软件生产周期中的一步，系统修改不论是通过拖拽界面还是高级编程语言来实现，反复地分析、测试、构建和部署还是必须的，并不会因此而消失。在评估一个零代码定制软件包时，要确保产品支持这些过程，并具有实施零代码定制软件包所需的IT支持。

语言

软件行业在编程语言方面正在经历一场复兴。ThoughtWorks认为现在是时候评估哪些语言对你的公司更有帮助，同时评估所做出选择的有效期限。你需要关注语言。有独立支持团队的传统结构化的组织可能会发现技术限制了选择，此时，DevOps提供了一个途径。

函数式编程的发展依然较为缓慢，但其在开发人员心中的地位和代码库却在稳步上升。Scala、Clojure、F#等新语言提供了许多新功能。现在Functional Java、TotallyLazy和LambdaJ等库将某些函数式语言的特性，尤其是一些高阶函数的特性和集合操作，移植回了Java。我们喜欢这种趋势，因为它展现出了未来语言的常见功能，并且还允许开发人员在自己熟悉的语言中使用这些新特性。

微软F#不断发展，最近发布了F# 3.0 beta。F#十分擅长清晰地表达业务逻辑和领域逻辑。想要在程序中清晰表达业务逻辑的开发人员，可以选择使用F#，而其他主要管道代码使用C#。

ClojureScript仅仅展示了Clojure核心的跨平台功能，它将Clojure的主要部分移植到了JavaScript上来运行，抛弃了一些Clojure在JVM和CLR上的独特功能，比如软件事务存储，但ClojureScript更高级的复杂功能却具有令人惊讶的高保真度。ClojureScript提供的一个有趣的功能是可以使用



图4 语言趋势雷达

ClojureScript将含有JSON格式的数据结构以原有的形式发送出去。这是因为Clojure是一种Lisp语言，也就意味着你可以发送“真正”的代码。

将丰富的用户体验通过网络传送到桌面、平板电脑和手机设备主要依赖于JavaScript，我们继续推荐在你的应用将JavaScript作为“一等公民”语言。开发人员应认真考虑如何采用与其他编程语言同样严格的标准来构建、测试、重构和维护JavaScript代码。

由于JavaScript语言本身的缺陷及其内在的性能问题，Google发明了Dart语言用来取代JavaScript。Dart与之前Google发明的语言一样，其语法和语义与Java类似，想要比JavaScript原型特性更具吸引力。浏览器开发社区对Dart的接受程度仍然比较低，这也是可以理解的，Chrome的持续发展和CoffeeScript等替代产品的研究可能会打破这种僵局，但该语言是否能被广泛接受还有待时间考验。

（感谢ThoughtWorks中国公司授权并翻译了本文，原文链接为http://www.thoughtworks.com/radar）

学院派的复兴

PostgreSQL和MySQL应用场景对比

文 / 萧少聪

近年来，由于开源社区的努力，以Hadoop、MangoDB为首的NoSQL技术迅速为国人所青睐，而传统关系型数据库领域则喜忧参半。在本文中，作者将结合自己的体会，分析最适合PostgreSQL的应用场景。

目前国内“骑”过PostgreSQL这只“大象”的用户寥寥可数，更不用提有能力“驯服”它的专家了。不过随着2010年“PostgreSQL中国用户组”的成立，越来越多技术同僚加入到我们的行列，成为开源数据库阵地的一员。

在中国，用户对MySQL的耳闻较多，这也确立了MySQL在数据库圈中的地位，而PostgreSQL爱好者则成了“异教徒”。我希望通过本文让更多技术同僚了解我们钟爱PostgreSQL的原因，希望能激发更多用户对PostgreSQL产生兴趣，继而进行研究，为PostgreSQL在国内的发展作一份贡献。

下面我将针对两大开源关系数据库产品MySQL及PostgreSQL进行比较，但请不要怀着竞争的心态，因为世界上从来没有最好的数据库，只有最适合目标应用场景的数据库。而实际上，我们不应该简单评判MySQL及PostgreSQL之间的优劣，因为这是两个设计思路及目标完全不同的数据库产品（想想看，海豚和大象即使在动物园也不可能打到一起）。

历史介绍

MySQL的历史

- 1995年3月，MySQL发布了第一版，提供了基本的SQL功能。
- 1999年MySQL AB公司成立，此后MySQL产品版本及社区维护都由该公司进行。
- 2005年初推出4.0.2版，支持事务操作并以LAMP架构风靡全球，成为“最流行的开源数据库”。
- 2008年2月，MySQL AB被Sun收购，同年作为MySQL核心开发人员之一的Brian Aker离开Sun公司并发布自己的分支版本Drizzle。
- 2009年1月，也就是Oracle收购Sun的前一年，原MySQL AB联合创始人Michael Widenius离开Sun，并于同年创立Monty Program AB，发布另一MySQL数据库分支MariaDB。
- 2010年1月，Sun被Oracle收购，MySQL也同时并入了Oracle的产品线。

自此，MySQL出现了3个不同的分支版本分别由Oracle、Monty Program AB及Drizzle社区维护。而国内使用量最大的依然是由Oracle维护的版本，此版本基于GPL协议，但有部分功能只能通过EULA协议包含在商业版本中。

PostgreSQL的历史

- 1975年，受到IBM System R项目的启发，Michael Stonebraker和Eugene Wong开始了Ingers

项目，该项目获得了自然科学基金和三个军事机构的资金支持。

- 1985年，Michael Stonebraker基于Ingres项目发起了Postgres（post-Ingers）项目，致力于解决关系模型中的数据库管理局现性问题。
- 1995年，Andrew Yu和Jolly Chen在Postgres中加入了SQL查询语言并更名为Postgres95。
- 1996年，Bruce Momjian等人重整源代码，发布了第一个正式的开源版本，命名为PostgreSQL 6.0，并于1999年开始支持MVCC多版本控制。
- 2000－2009年，相继发布了7.x及8.x等多个版本，并在功能及性能上进行了重大改良与突破，由于一直遵循学院派“以功能至上”的研究精神，被誉为“全球最先进的开源数据库”。
- 2010年，9.x版本发布，重点增强了数据库集群备份、异构外部数据操作、数据安全性，并取得了重大性能提升。

PostgreSQL基于自己的开源协议，类似BSD和MIT协议，它的开放性很强，不限定在其基础上开发的代码是否要继续开源。因此，不少数据库都基于它进行二次开发，其中以Greenplum最为知名，它针对大数据应用提供高性能的分布式列式数据库；其次是EnterpriseDB，它针对企业级OLTP在线事务处理系统提供Oracle兼容性（其引擎还被嵌入到IBM DB2数据库中）、高速无限缓存技术并兼容PostgreSQL的管理工具。国产数据库中也不乏基于PostgreSQL开发的作品。

MySQL Vs. PostgreSQL：历史和未来

对比两者的历史可知，PostgreSQL数据库历史更为悠久，但MySQL在本世纪之初的爆炸性网络发展中获得了巨大的发展，而PostgreSQL则在此时默默地坚守学院派以功能为核心的研究之路。MySQL最早的历史其实可以追溯到1979年，但从功能上看，当时只是一个面向报表的存储引擎，名为“Unireg”。因此，直到1995年MySQL第一版推出，才可以真正称作关系型数据库。

在中国，互联网发展过程中也积累了大量MySQL人才和丰富的经验；而PostgreSQL则因其一直注重对功能及完整性要求苛刻的OLTP在线事务处理应用，而败给了盗版应用猖獗的商业数据库。从现有的统计数据看来，两者的使用数量差距，可能达到数十倍之多。

从社区发展角度来看，PostgreSQL与Linux有很多相似之处：独立社区不属于任何企业，由众多企业进行支持及赞助；同时衍生出商业的发行版本进行商业运作，从而推动社区的持续发展及市场地位，厚积薄发。由于MySQL近年发展动荡，不少知名企业纷纷开始计划使用PostgreSQL对其进行替代，Apple在最新的Mac OS X 10.7中已经完成了此举。当然也有那些拥有大量IT大牛的技术型企业，在节约成本过程中继续选择使用MySQL数据库，并通过基于全源代码编译的安装方式以免除法律风险。在国内，淘宝网就是很好的例子。

MySQL拥有扎实的群众基础，相信无论经历多大的风雨，只要有基于开源协议的版本，都将获得广大技术人员的支持和拥护。对于PostgreSQL，在中国版权意识不断提高，以及法律法规不断完善的大环境下，未来的发展潜力也不可忽视。

特性功能分析

下面我将针对技术人员关心的几个方面，对两种数据库的特性进行分析。

性能

MySQL当前使用较多的有两个引擎，一是MyISAM，另一个是InnoDB。总体来说，MyISAM引擎在非事务操作环境及非并发环境中性能十分强劲，我想即使Oracle也不是它的对手 。但如果是在高并发环境中，由于MyISAM使用的锁机制为“表级锁”，就显得力不从心了；此外MyISAM不支持事务操作，因此无法处理事务环境。InnoDB引擎支持事务及“行级锁”，因此在高并发事务处理环境下性能优于MyISAM，在不少测试中接近了Oracle的水平。但由于MySQL本身基于线程结构，在x86服务器已经踏入到多路多核时代的今天，并没有发挥好CPU的性能。为

了解决这个问题，Google技术团队在2010年发布了SmpPerformance补丁包，令情况暂时有所好转，然而在CPU核心快速增长的趋势下，还需要从“根”上解决问题。另一方面，在大数据量的处理能力上，不少用户指出单表超过5000万行后性能开始下降，但我认为可以通过架构调整和系统优化进行解决。

PostgreSQL天生遵循ACID原则，因此如果要对比非事务处理性能，那是无能为力了。如果你对PostgreSQL的早期性能进行过了解，相信都会对VACUUM心有余悸，在7.x甚至到8.x中它都是DBA的痛中之痛，甚至烙下了“PostgreSQL不适合7×24小时运作”的骂名。自8.0版以后，PostgreSQL提供了AutoVACUUM，可以对所有表进行自动VACUUM清理及ANALYZE分析，但根据我在不同版本中的使用体会，这一功能直到9.0以后才真发挥了成效。VACUUM是把双刃剑——每次执行都会对系统空间进行释放，并可结合ANALYZE为SQL解析器进行公式优化，以提升对该表操作时的性能；但另一方面，每次执行又都会耗费服务器的处理能力。9.0以后，AutoVACUUM得到了很大改进，我在最新的9.1版中对单表（3亿行，20字段，3个Index）进行16个进程的同时Insert及Select（包括Sum及基于Index的group by）压力测试，Insert可以稳定在60000行/秒，Select保持在0.85秒左右。

易用性

从安装角度来看，两个数据库都提供源码编译安装、RPM包安装、图形化安装三种方式。对于初学者，我推荐图形化安装，以减少出错，同时可以提高系统的稳定性。因为图形安装方式会针对不同操作系统带有比较全面的连接库，不用理会连接库兼容性问题。

从学习的角度来看，MySQL容易上手，市场上参考资料很多，不少同学在学校就已经接触过，因此在易用性方面，MySQL在开源数据库领域中排行第一。唯一的缺陷就是官方文档不太好用，很多问题无法从官方文档中快速找到答案，但也许这只是我的个人感受， 只是因为我对这个风格的文档不太适应罢了。而PostgreSQL的官方文档做得相当出色，圈子里不少朋友都是直接通过阅读官方文档开始学习的。但PostgreSQL的参考资料比较少，特别是中文资料，目前国内只有一本分析PostgreSQL内核的中文书籍，很多信息都需要通过网络进行获取。

从操作角度来看，两者没有太大的差别，都有命令行工具、图形工具及基于Web的管理工具。

因此从学习角度来说，MySQL占尽天时、地利、人和，而对于需要进行深入研究的应用，我估计两个数据库的难度差不多。为了解决PostgreSQL学习资料匮乏的问题，我正与PostgreSQL用户会的几位同僚筹备编写针对DBA及开发人员的书籍，希望在不久的将来可以与大家见面。

功能（MySQL 5.5对比PostgreSQL 9.1）

针对一般的数据库需求，这两个数据库都可以很好地胜任，而深入的功能方面我认为PostgreSQL更有优势，因为这是它的传统强项。PostgreSQL拥有以下特点。

- 丰富的数据类型：除传统数据类型外，还支持网络数据类型、ltree、地理信息类型、UUID等。举个例子，如果你希望做一个商品系统，需要存放大量类似“电子产品”→“电脑”→“平板电脑”→“苹果”这样的信息，使用ltree类型可以大大提高查询导航信息时的性能。

- 丰富的索引类型：除了常用的索引类型，还可以支持GiST及GIN索引，以上所提到的ltree就基于GiST索引，你也可以灵活定义自己的数据类型，并为它建立独特的索引以提高处理速度。另外PostgreSQL还支持函数索引及部分索引，可以在某些场景下大幅提高性能。

- Unlogged Tables：支持在CREATE TABLE时指定某个数据表不写入LOG日志。经测试，写入性能可以提高近3倍，但如果数据库系统故障崩溃，此表将会丢失部分数据。

- 在线表结构调整：支持在运行过程中对表结构进行在线修改，包括添加、删除、修改字段，而MySQL只支持添加字段。

■ 触发器&存储过程：PostgreSQL的触发器支持基于表级别触发及字段级别触发，而MySQL只支持表级别；PostgreSQL的存储过程相对MySQL更为完善，并可通过pgAdmin的Debugger插件进行如同高级语言一样的单步调试。另外可以通过Java、C等语言扩充存储过程，完成特殊的功能，比如通过存储过程直接实现邮件发送或端口调用。

■ SQL/MED：支持将数据存放到数据库以外的CSV文件中作为外部表使用，通过不同的FWD扩展还可以支持与Oracle、MySQL、LDAP甚至Hadoop等不同数据源的连接。

■ Join Bitmap索引：虽然PostgreSQL不支持指定类型的Bitmap索引，但在进行Join操作时系统会通过优化机制，触发基于内存的Bitmap索引以提高查询性能。

■ Window函数：如果你有各种统计需求，或者已经在使用Oracle及一些高级数据库，相信对Window函数不会陌生，如OVER (PARTITION BY xxx)可以基于xxx字段进行分类汇总。这个功能对于复杂的报表统计十分重要，否则我们就要编写大量性能低下的子查询或通过应用程序实现了。

应用场景对比

Web网站

推荐使用MySQL。通常企业网站的需求比较简单，而且不牵涉到复杂数据库功能，同时已经有很多现成的模版及系统，通过LAMP架构绝对是最佳组合。

Web交易平台

推荐使用PostgreSQL。此类网站在发展之初可能功能及压力都有限，无需特意选择。我们可以看到MySQL在功能及高并发性能上也一直在努力，在国内Web交易平台的老大淘宝网已经对MySQL进行了大量投入，甚至是对MySQL的内核进行Patch以提高它在特定环境中的稳定性，但这些Patch是否会全数公开我们还不得而知。因此，考虑到长远的发展过程及产品成熟度，我依然推荐使用PostgreSQL，这样可以更好地应对未来对高级功能特性的需求及高并发增长。

在线游戏

推荐使用PostgreSQL。在线游戏环境中的实时并发压力很大，PostgreSQL的多线程设计可以很好地支持这类应用环境，当服务器性能不足时还可以通过多种方案实现数据库的分布式扩展。杭州思凯网络就是很好的例子，2011年的PostgreSQL中国用户组大会中，周正中曾经就此话题做过十分精彩的分享，有兴趣的读者可以在网上搜索这场演讲的PPT。

企业周边系统（Email、门禁、监控等）

推荐使用MySQL。此类系统实时压力及数据量有限，对于稳定性的要求也不高，以Email系统为例，数据库基本只用于邮件用户名的查询，收发邮件时邮件内容信息与数据库无关。国内开源系统中使用Postfix+MySQL+ExtMail的用户众多，系统都运行得十分顺畅。

企业重要系统（流程管理、生产环节管理系统、CRM系统）

推荐使用PostgreSQL。随着企业的发展，此类系统的压力可能在一两年内遇到突发性的增长。当企业达到一定规模后，对于报表及实时汇总的需求会越来越复杂，Window函数则可以很好地解决此问题。另一方面，企业系统众多，或许将来还会引入第三方系统，在对接过程中，SQL/MED将为你提供更为便捷的通道。

企业核心系统（ERP系统、财务系统、BOSS系统）

推荐沿用现有数据库。同时我也推荐从零搭建这类系统时选择PostgreSQL，因为9.0版后，它的功能、性能已经可以胜任这类系统的需求，特别是9.1版还支持同步数据复制，已经可以满足对生产环境99.999%的高可用性要求。但此类系统牵涉到企业的生命线，如果企业直接使用没有商业服务支持的PostgreSQL系统，对于生产环境来说有很大风险，因此如果要在企业核心系统中使用PostgreSQL，就需要企业自身拥有相应的DBA技

术团队以实现自服务，或寻求拥有丰富经验的服务机构进行技术支持。

小结

综合评价

■ MySQL：以易用及够用为理念，经过多年升级，已经从最初只提供简单的存储结构，发展出后来的外键支持、事务处理支持、行级锁、数据复制等高级功能。因此十分适合各种应用的快速开发。另一个值得称赞的特点是，MySQL支持不同存储引擎在同一数据库中的混合使用，以适应不同场景对数据表性能及功能的使用需求。借助20世纪90年代金融危机前后，全球经济不景气的大环境，在互联网大爆发的过程中赢得了重大发展机遇。免费、轻量、易于使用、架构灵活，在企业发展的过程中，MySQL可谓是IT数据库系统的绝佳选择。然而MySQL在触发器、存储过程、高并发性能的表现上却稍逊一筹，因此在很多大中型企业架构中，无法应对企业的复杂需求。

■ PostgreSQL：从设计之初就以关系模型为研究对象，由学院派主导着发展思路。在6.x~7.x的发展过程中十分保守，加之当时未有多核CPU，多线程的设计在x86服务器中无法得到发挥，而错过了基于x86架构快速发展的互联网爆发黄金期。8.x~9.x得益于多核CPU的发展及本身对内核性能的大量优化工作，在高并发的事务环境下性能表现卓著。在2007年的金融危机后，不少企业由于成本压力，开始考虑将企业重要业务从昂贵的垄断厂商中释放出来。借助长期从学院派发展而来的强大功能，PostgreSQL获得了越来越多青睐，当然这个过程相比周边系统的迁移要漫长得多。但当前PostgreSQL的核心问题是中文资料及技术人才缺乏，最严重还的是在技术人才方面高、中、低三个等级之间存在明显的断层。

通过以上介绍、分析和对比，读者应该已经对两个数据库的优劣及在不同场景中的适用性有了较为全面的了解。相信到这里，您可能有了以下一些想法，且看我猜得是否正确。

■ 不考虑使用PostgreSQL：或许曾经由于传闻中PostgreSQL的功能及未来发展，而计划将现有系统从MySQL迁移到PostgreSQL，但看完此文后发现，这些功能对现有系统都没实质用处，MySQL已经符合现有环境甚至未来的发展。

■ 考虑使用PostgreSQL：由于MySQL更简易，用户量更多、企业应用成本相对更低，因此使用MySQL的计划不会改变；但在高技术要求的系统中，已经开始认真考虑从昂贵的商业数据库迁移到PostgreSQL的可行性，这可能为企业节省大量的成本并避免未来的法律风险。

■ 学习永远是美好的：作为一名程序员，工作中只是直接使用数据库，由于系统通过ORM工具进行连接，用什么数据库都对开发没有太大影响，但多了解不同数据库的优势和缺点，对未来职业发展有益无害。

当然如果你也和我一样， 是一名PostgreSQL狂热分子，就让我们一同为这只敦厚、纯朴的“大象”挺身而出，为 PostgreSQL在中国的发展社区一同分享和贡献。PostgreSQL中国用户组，也期待你的参与。



萧少聪

北京神州立诚科技有限公司技术总监，2011年参与筹建PostgreSQL中国用户组。2010年与何世晓、杜朝晖等人共同编写了《Linux系统案例精解》。

责任编辑：卢鹄翔（ludx@csdn.net）

Ready? Go!

Go语言并发模型和工具链

文 / 邓楠

在《程序员》5月刊中，作者介绍了Go语言的部分语法和类型系统，本文作为延续，将重点介绍Go语言的并发模型和工具链。

并行和goroutine

然而，处理器技术的发展指出，比起单处理器，由多个类似的处理器组成的多处理器计算机也许会更加强大、可靠和经济。

——C.A.R. Hoare，CSP作者，写于1978年

20世纪六七十年代，为了弥补处理器的计算能力不足，并行计算曾一度成为研究热点。期间不乏优秀的想法，如信号量、管程、锁以及基于消息传递的同步机制。但八十年代起，随着单核处理器性能飞速提高，学术界迎来了并行计算的黑暗时期。六七十年代的研究成果中，只有早期的一些思想被大规模使用在实际开发中。而七十年代后期的很多成果甚至还没被大规模应用，就伴随着并行计算黑暗期的到来，或不温不火，或被收藏入库，CSP（Communicating Sequential Processes）便是其中之一。但它优雅简洁的处理方式依然在一些小众语言中流传了下来。如今，由于能耗和散热问题，处理器的发展转而以多核的方式提高性能，我们再次迎来了曾经面对过的并行计算。这时候，CSP模型逐渐展露头脚。

CSP的基本思路是基于消息机制的同步和数据共享。与传统的锁同步不同，消息机制简化了程序设计，并且可以有效地减少潜在Bug。基于CSP模型的语言主要有三个分支：忠于原始CSP设计，以Occam为代表；强调网络和模式，以Erlang为代表；再一个就是强调传递消息的信道，以Squeak、Newsqueak、Alef、Limbo和Go为代表。

既然说起Go的这一分支是以强调信道为特色，那么就先从Go的信道说起。Go的信道是一种数据类型，goroutine可以使用它来传递数据。至于goroutine是什么，后文会详细讨论。此处仅需把它理解为与线程类似的运行时结构即可。

定义一个信道，需要指定这个信道上传递的数据类型。可以是int、float32、float64等基本数据类型，也可以是用户自定义的结构体、接口、甚至可以是信道本身。

```
ch := make(chan int)
```

这样，就定义了一个传递整数类型的信道。如果要从这个信道中读取一个值，则可以使用<-操作。类似的，写入则使用->操作符：

```
// 从ch中读取一个值存入i中
i := <- ch

// 向ch中写入j的值
ch <- j
```

信道的操作是同步的，一个读操作只有在真正读到内容之后，才继续执行下面的语句；而写操作

则只有在写入数据被信道另一端读到，才执行之后的语句（Go中信道也可以加入缓存队列，在此不多讨论）。

同时，对于信道还允许使用for循环依次处理来自信道的内容：

```
func handle(queue chan *Request) {
    for r := range queue {
        process(r)
    }
}
```

这个函数的任务就是不断地从信道中读取Request结构体的指针，然后调用process函数进行处理。

除此以外，还可以使用select对多个信道进行读写操作：

```
func Serve(queue chan *Request,
           quit chan bool) {
    for {
        select {
        case req := <- queue:
            process(r)
        case <- quit:
            return
        }
    }
}
```

这个函数接受两个信道作为参数。第一个信道queue用来传递各种请求。第二个信道quit则用来发布一条信令，告诉该函数返回。

接下来要说的，就是goroutine。它是一种比线程还要轻量的并行结构。在Go程序运行时，一般会并行运行几个线程，然后把goroutine分配到各个线程中。当一个goroutine结束或者被阻塞时，另外一个goroutine将被调度到被阻塞或结束的goroutine所在的线程中。这样的调度保证了每个线程可以有较高的使用率，不必一直处于阻塞状态。由此省去了很多操作系统调度线程而导致的上下文切换。按照Go官方的说法，一个Go程序同时运行几万到几十万个goroutine是非常正常的。

使用一个goroutine也非常简单，只要在函数调用前面加入go就可以了。

```
go process(r)
```

这样，process这个函数就单独运行在一个goroutine中了。

由此带来的结果，就是极度简化了服务器端对并发连接的处理。众所周知，如果让一个线程只处理一个用户连接，那么开发起来会非常简单，但是效率不高；而如果一个线程处理多个用户连接，又无端增加了开发难度。而配合信道使用goroutine则在不增加开发难度的同时，也提高了效率。

考虑这样一个应用场景：服务器从网络接收客户端请求，做一些处理，再把结果返回给客户。

对于不同的用户连接，用不同的goroutine处理。定义名为UserConn的结构体来表示一个用户连接。同时，这个结构体定义了一个叫做ReadRequest的方法，用于从网络读取用户的请求；还有一个叫做WriteResponse的方法，用于从网络给用户传递结果。作为一个想象的例子，具体的实现细节在此不详述。

那么，对于每个连接，要做的事情大约如此：

```
func ServeClient(conn *UserConn) {
    ch := make(chan *Response)

    // 创建一个goroutine，
    // 专门用于向用户发送结果
    go writeRes(conn, ch)

    for {
        // 读取一个请求，
        //  判断类型
        // 如果用户请求关闭，
        //  则函数返回
        req := conn.ReadRequest()
        switch req.Type {
        case NORMAL_REQUEST:
            go process(req, ch)
        case EXIT:
            return
        }
    }
}
```

writeResponses和process的基本结构大约如下：

```
func writeRes(conn *UserConn,
              ch chan *Response) {
    for r := range ch {
        conn.WriteResponse(r)
    }
}

func process(req *Request,
             ch chan *Response) {
    res := calculate(req)
    ch <-res
}
```

信道本身很符合人们对于通信工具的直觉定义，

开发者可以很自然地使用信道在goroutine之间建立各种关系。使用信道和goroutine，每个函数要完成的任务都被单一化，减少了发生错误的可能。代码中，通过传递指针的方式来共享内存空间，在每次共享之前，都是以消息进行同步。这又是一条Go的原则：用传递消息来共享内存，而不是用共享内存来传递消息。由此简化了并行程序的开发。

作为一门实用的编程语言，Go并没有按照CSP原始论文中说的，仅仅提供信道的方式来进行同步。Go在标准库中也提供了基于锁、信号量等传统同步机制的工具。在以上代码中，其实存在着一个潜在Bug：ServeClient函数不是在所有运行process的goroutine执行结束后再退出，而是在一收到来自客户端的退出命令后直接退出的。更合理的操作应该在所有处理该连接的goroutine都退出后再返回。在标准库中，有一个`WaitGroup`结构体就可以专门解决等待多个goroutine的问题。

接下来，为每个用户连接开启一个goroutine，执行`ServeClient`函数。前面已经说过，由于goroutine是一种比线程还轻量的调度单位，如此数目的goroutine并不会带来严重的性能下降。

由于goroutine和消息机制简化了开发，并且Go也鼓励这样的设计，开发者会自觉地选择基于多个goroutine的设计。由此带来了另一个好处，就是程序在多核系统上的扩展性。随着处理器核心数量的增加，如何发掘程序内在的并行结构成了当前开发人员面临的挑战。而使用Go编写、基于多个goroutine的设计，往往会天生具备着足够的并行结构来扩展到多核处理器之上。每个goroutine实际都是可以放在一个独立的处理器上，与其他goroutine并行执行。也就是说，今天为4核处理器写的代码，也许不必修改，就可以运行在未来128核的CPU上，并且同时使用所有的内核。

无需配置，直接编译

如果Go需要一个配置文件，描述如何编译和构建Go写的程序，那就是Go的失败。

——Go官方文档

对于make、autoconf、automake等用于指定编译顺序和依赖关系的工具，Go的态度是：开发者在写代码时，就留下了关于依赖的足够信息，不该要求开发者再单独写一份配置文件，去指明依赖关系和编译顺序。为此，开发者只需要在安装Go工具链之后，按照官方文档，配置好一个目录结构和一个环境变量即可。以后安装任何Go程序，编译任何Go程序、库都只需要几条简单的命令就可以了。

对于一个自包含（不依赖任何第三方库）的程序，只需要在当前目录下运行`go build`就会编译好整个程序。

如果程序依赖第三方库，又该如何呢？很简单，在代码中的import语句里，写入第三方库的在网络中的位置即可。这里的import和Java、Python中的import的概念一样，都是引入一个包。

```
import (
    "fmt"
    "github.com/monnand/redis.go"
)
```

import中引入的第一个包是fmt，这是标准库中的包，提供Printf一类的格式化输入和输出。第二个引入的包则是位于Github上的代码库。它会引入Github用户monnand下的redis.go这个项目定义的包。接下来，再调用go命令安装这个库：

```
go get github.com/monnand/redis.go
```

这样，Go程序就会自动下载、编译和安装这个库（包括它的依赖）。接下来再使用`go build`编译依赖redis.go的程序。

除此以外，如果依赖redis.go的程序也在github（或其他Go支持的版本控制库）中，那么只用一条`go get`命令指明该程序所在的远程地址就足够了，go会自己下载安装各种依赖。除了Github，Go还支持Google Code、BitBucket、Launchpad，或者是任何位于其他服务器上，使用svn、git、Bazzar、Mercurial做版本控制的Go程序和库。这一切都极大地简化了开发人员和最终用户的操作。

再谈运行效率

Matt：使用Pat/Go后，比起Sinatra/Ruby方案，JSON API节点效率提升了多少？给个估计就可以。

Blake：大约10,000倍。

Matt：漂亮！我能引述你的话吗？

Blake：我再查查，我觉得好像低估了。

——M. Aimonetti与B. Mizerany在Twitter上的对话

Go程序的运行效率一直是人们关注的焦点。一方面，Go的语法、类型系统都非常简单，为编译器的开发和优化提供了很大空间；另一方面，Go作为静态编译型语言，代码直接编译为机器码，无需中间解释。

不过倘若在网上搜索一下，就会发现关于Go程序的运行效率存在着严重的两极分化。一部分测试显示，Go的程序运行效率非常高，甚至一些方面超过了C++写的同等程序；另一部分测试则显示，某些方面，Go甚至不如Stackless Python写的脚本。

Go编译器本身虽然还存在很大优化空间，但产生的机器码效率已经比较高。而标准库（其中包括各种运行时代码，比如垃圾回收、哈希表等）还没有怎么优化，甚至有些还处于很初级的阶段。这是网络上的测试结果存在着严重差异的原因之一。另外，作为一门新语言，开发人员由于对它不熟悉，写出的代码可能存在性能瓶颈，也加大了评测结果的差异。

Go语言的开发者之一，Russ Cox曾在Go的官方博客上发表了一篇文章。其中使用了某基准测试程序（Benchmark）的代码，分别优化了其中的C++测试和Go测试部分。优化后的Go程序运行时间，甚至仅仅是优化后的C++程序运行时间的65.8%！这也从一个侧面反应出了Go的潜力。

当前Go语言中，还存在不少缺陷：垃圾回收还处于比较初级的阶段，而且对于32位系统的支持还不太完善，一些标准库的代码还有待优化。按照Go官方的说法，未来将会使用完全并行的垃圾回收器，这对于性能来说将会有很大的提高。而随着Go 1的发布，Go开发组也会将精力从语法和标准库的规范，转移到对编译器和标准库的优化上。Go程序的运行效率，目标将会是逼近C++，超越Java。

总结

当前，我觉得在系统级开发方面，它（Go）比C++要好上许多。使用它开发更高效，并能使用比C++更简单的方式解决很多问题。

——Bruce Eckel，《C++编程思想》作者

UNIX创始人Ken Thonpson，UNIX/Plan 9开发者Rob Pike、Russ Cox，Memcached作者Brad Fitzpatrick，Java Hotspot编译器、Chrome V8引擎作者之一Robert Griesemer，Gold连接器作者、GCC社区活跃开发人员Ian LanceTaylor……当这样一群人凑在一起，无论开发什么，这团队本身也许已经足以吸引众人眼球了。而Go作为这样一门团队开发出的语言，目前为止还是给不少人带来了惊喜。

已经有很多公司使用Go开发生产级程序。Rob Pike曾透露过Google内部正逐渐开始使用Go，YouTube则使用Go编写核心部件，并且将部分代码组织成了开源项目vitess。国内包括豆瓣、QBox等公司也已经率先踏入Go语言这个领域。

随着Go 1的推出，一个稳定的Go语言平台和开源社区已经形成。对于喜欢尝试新鲜语言的开发者来说，Go不失为一个选择。



邓楠

自由/开源软件支持者，GNU.org中文翻译组组长。一年多以前开始学习Go，目前正在使用Go语言开发一个开源项目http://uniqush.org。

责任编辑：卢鹄翔（ludx@csdn.net）

稳定婚姻问题和Gale-Shapley算法

文 / 顾森

什么是算法？每当有人问作者这样的问题时，他总会引用这个例子：假如你是一个媒人，有若干个单身男子登门求助，还有同样多的单身女子也前来征婚。如果你已经知道这些女孩儿在每个男孩儿心目中的排名，以及男孩儿们在每个女孩儿心中的排名，你应该怎样为他们牵线配对呢？

最好的配对方案当然是，每个人的另一半正好都是自己的“第一选择”。这虽然很完美，但绝大多数情况下都不可能实现。比方说，男1号最喜欢的是女1号，而女1号的最爱不是男1号，这两个人的最佳选择就不可能被同时满足。如果好几个男孩儿最喜欢的都是同一个女孩儿，这几个男孩儿的首选也不会同时得到满足。当这种最为理想的配对方案无法实现时，怎样的配对方案才能令人满意呢？

其实，找的对象太完美不见得是好事儿，和谐才是婚姻的关键。如果男1号和女1号各有各的对象，但男1号觉得，比起自己现在的，女1号更好一些；女1号也发现，在自己心目中，男1号的排名比现男友更靠前。这样一来，这两人就可能抛弃各自现在的对象——如果出现了这种情况，我们就说婚姻搭配是不稳定的。作为一个红娘，你深知，对象介绍得不好没关系，就怕婚姻关系不稳定。给客户牵线配对时，虽然不能让每个人都得到最满意的，但搭配必须得稳定。换句话说，对于每一个人，在他心目中比他当前伴侣更好的异性，都不会认为他也是一个更好的选择。现在，我们的问题是：稳定的婚姻搭配总是存在吗？应该怎样寻找？

一次失败的尝试

为了便于分析，我们下面做一些约定。我们用字母A、B、C对男性进行编号，用数字1、2、3对女性进行编号。我们把所有男性从上到下列在左侧，括号里的数字表示每个人心目中对所有女性的排名；再把所有女性列在右侧，用括号里的字母表示她们对男性的偏好。图1所示的就是2男2女的一种情形，每个男的都更喜欢女1号，但女1号更喜欢男B，女2号更喜欢男A。若按A-1、B-2进行搭配，则男B和女1都更喜欢对方一些，这样的婚姻搭配就是不稳定的。但若换一种搭配方案（如图2），这样的搭配就是稳定的了。



图1　一个不稳定的婚姻搭配



图2　一个稳定的婚姻搭配

可能很多人会立即想到一种寻找稳定婚姻搭配的策略：不断修补当前搭配方案。如果两个人互相都觉得对方比自己当前的伴侣更好，就让这两个人成为一对，剩下被甩的那两个人组成一对。如果还有想要私奔的男女对，就继续按照他们的愿

望对换情侣，直到最终消除所有的不稳定组合。容易看出，应用这种“修补策略”所得到的最终结果一定满足稳定性，但这种策略的问题在于，它不一定存在“最终结果”。事实上，按照上述方法反复调整搭配方案，最终可能会陷入一个死循环，因此该策略甚至不能保证得出一个确定的方案来，如图3所示。

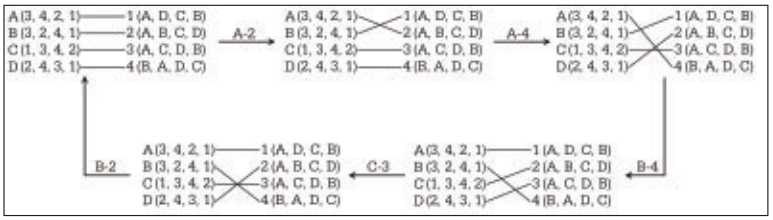


图3　应用“修补策略”可能会产生死循环

Gale-Shapley算法

1962年，美国数学家David Gale和Lloyd Shapley发明了一种寻找稳定婚姻的策略。不管男女各有多少人，也不管他们的偏好如何，应用这种策略后总能得到一个稳定的搭配。换句话说，他们证明了稳定的婚姻搭配总是存在的。有趣的是，这种策略反映了现实生活中的很多真实情况。

在这种策略中，男孩儿将一轮一轮地去追求他中意的女子，女子可以选择接受或者拒绝他的追求者。第一轮，每个男孩儿都选择自己名单上排在首位的女孩儿，并向她表白。此时，一个女孩儿可能面对的情况有三种：没有人跟她表白，只有一个人跟她表白，有不止一个人跟她表白。在第一种情况下，这个女孩儿什么都不用做，只需要继续等待；在第二种情况下，接受那个人的表白，答应暂时和他做情侣；在第三种情况下，从所有追求者中选择自己最中意的那一位，答应和他暂时做情侣，并拒绝所有其他追求者。

第一轮结束后，有些男孩儿已经有女朋友了，有些男孩儿仍然是单身。在第二轮追女行动中，每个单身男孩儿都从所有还没拒绝过他的女孩儿中选出自己最中意的那一个，并向她表白，不管她现在是否是单身。和第一轮一样，女孩儿们需要从表白者中选择最中意的一位，拒绝其他追求者。注意，如果这个女孩儿已经有男朋友了，当她遇到了更好的追求者时，她必须拒绝掉现在的男友，投向新的追求者的怀抱。这样，一些单身男孩儿将会得到女友，那些已经有了女友的人也可能重新变成光棍。在以后的每一轮中，单身男孩儿继续追求列表中的下一个女孩儿，女孩儿则从包括现男友在内的所有追求者中选择最好的一个，并对其他人说不。这样一轮一轮地进行下去，直到某个时候所有人都不再单身，下一轮将不会有任何新的表白发生，整个过程自动结束。此时的婚姻搭配就一定是稳定的了。

这个策略会不会像之前的修补法一样，出现永远也无法终止的情况呢？不会。下面我们将说明，随着轮数的增加，总有一个时候所有人都能配对。由于在每一轮中，至少会有一个男孩儿向某个女孩儿告白，因此总的告白次数将随着轮数的增加而增加。倘若整个流程一直没有因所有人都配上对了而结束，最终必然会出现某个男孩儿追遍了所有女孩儿的情况。而一个女孩儿只要被人追过一次，以后就不可能再单身了。既然所有女孩儿都被这个男孩儿追过，就说明所有女孩儿现在都不是单身，也就是说此时所有人都已配对。

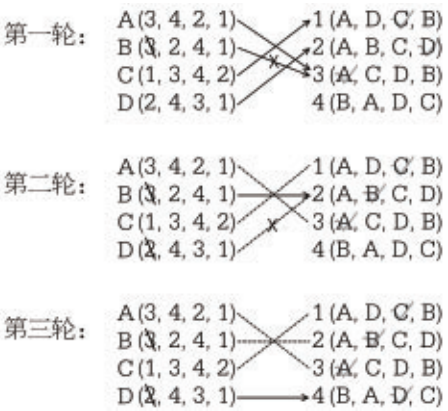


图4　应用上述策略，三轮之后将得出稳定的婚姻搭配

接下来，我们还需要证明，这样得出的配对方案确实是稳定的。首先注意到，随着轮数的增加，一个男孩儿追求的对象总是越来越糟，而一个女孩儿的男友只可能变得越来越好。假设男A和女1各自有各自的对象，但比起现在的对象，男A更喜欢女1。因此，男A之前肯定已经跟女1表白过。既然女1最后没有跟男A在一起，说明女1拒绝了男A，也就是说她有了比男A更好的男孩儿。这就证明了，两个人虽然不是一对，但都觉得对方比自己现在的伴侣好，这样的情况绝不可能发生。

我们把用来解决某种问题的一个策略，或者说一个方案，或者说一个处理过程，或者说一系列操作规则，或者更贴切地说，一套计算方法，叫做“算法”。上面这个用来寻找稳定婚姻的策略就叫做“Gale-Shapley算法”，有些人也管它叫“延迟认可算法”。

Gale-Shapley算法的意义和应用

每个算法都有它的实际意义，能给我们带来很多启发。Gale-Shapley算法最大的意义就在于，作为一个为这些男女牵线的媒人，你并不需要亲自计算稳定婚姻匹配，甚至根本不需要了解每个人的偏好，只需要按照这个算法组织一个男女配对活动就可以了。你需要做的仅仅是把算法流程当作游戏规则告诉大家，游戏结束后会自动得到一个大家都满意的婚姻匹配。整个算法可以简单地描述为：每个人都去做自己想做的事情。对于男性来说，从最喜欢的女孩儿开始追起是顺理成章的事；对于女性来说，不断选择最好的男孩儿也正好符合她的利益。因此，大家会自动遵守游戏规则，不用担心有人虚报自己的偏好。

历史上，这样的“配对游戏”还真有过实际应用，并且更有意思的是，这个算法的应用居然比算法本身的提出还早10年。早在1952年，美国就开始用这种办法给医学院的学生安排工作，这被称之为“全国住院医师配对项目”。配对的基本流程就是，各医院从尚未拒绝这一职位的医学院学生中选出最佳人选并发送聘用通知，当学生收到来自各医院的聘用通知后，系统会根据他所填写的意愿表自动将其分配到意愿最高的职位，并拒绝掉其他的职位。如此反复，直到每个学生都分配到了工作。那时人们并不知道这样的流程可以保证工作分配的稳定性，只是凭直觉认为这是很合理的。直到10年之后，Gale和Shapley才系统地研究了这个流程，提出了稳定婚姻问题，并证明了这个算法的正确性。

这个算法还有很多有趣的性质。比如说，大家可能会想，这种男追女女拒男的方案对男性更有利还是对女性更有利呢？答案是，这种方案对男性更有利。事实上，稳定婚姻搭配往往不止一种，然而上述算法的结果可以保证，每一位男性得到的伴侣都是所有可能的稳定婚姻搭配方案中最理想的，同时每一位女性得到的伴侣都是所有可能的稳定婚姻搭配方案中最差的。受篇幅限制，我们略去证明的过程。

这个算法会有一些潜在的问题。刚才我们已经说了，对于每位女性来说，得到的结果都是所有可能的稳定搭配中最差的一种。此时，倘若有某位女性知道所有其他人的偏好列表，经过精心计算，她有可能发现，故意拒绝掉本不该拒绝的人（暂时保留一个较差的人在身边），或许有机会等来更好的结果。因而，在实际生活中应用这种算法，不得不考虑一些可能的欺诈与博弈。

这个算法还有一些局限。例如，它无法处理2n个人不分男女的稳定搭配问题。一个简单的应用场景便是宿舍分配问题：假设每个宿舍住两个人，已知2n个学生中每一个学生对其余2n-1个学生的偏好评价，如何寻找一个稳定的宿舍分配？此时，Gale-Shapley算法就不再有用武之地了。而事实上，宿舍分配问题中很可能根本就不存在稳定的搭配。例如，有A、B、C、D四个人，其中A把B排在第一，B把C排在第一，C把A排在第一，而且他们三人都把D排在最后。容易看出，此时一定不存在稳定的宿舍分配方案。倘若A、D同宿舍，B、C同宿舍，那么C会认为A是更好的室友（因为C把A排在了第一），同时A会认为C是更好的室友（因为他把D排在了最后）。同理，B、D同宿舍或者C、D同宿舍也都是不行的，因而稳定的宿舍分配是不存在的。此时，重新定义宿舍分配的优劣性便是一个更为基本的问题。

稳定婚姻问题还有很多其他的变种，有些问题甚至是NP完全问题，至今仍然没有（也不大可能有）一种有效的算法。在图论、算法和博弈论中，这都是非常有趣的话题。



顾森

网名Matrix67，北京大学中文系应用语言学专业学生，数学爱好者。2005年开办数学博客http://www.matrix67.com，至今已积累上千篇文章，已有上万人订阅。

责任编辑：卢鹃翔（ludx@csdn.net）

1024核CPU上的并行编程

文 / 陈冠诚

摩尔定律在多核时代的表现形式是，内核数量几乎也会以成倍的速度增加，但内核数量的增长是否能与性能的增强划上等号呢？本文将探讨在这种背景下，如何编写程序才更有效。

让我们来看一个例子。假设一个程序80%的代码能被成功并行化，而剩下的20%只能串行执行，那么在未来的1024核CPU上，这个程序的加速比将会是：Speedup = 1 / (0.2 + 0.8/1024) = 4.98，即这个程序最多只能拥有5倍的加速比，哪怕是在1024核这样的“超级多核”CPU上，这就是著名的阿姆达尔定律。

难道在1024核CPU上我们不能把一个程序加速1024倍吗？当然可以，如果一个程序所有的代码都能被成功并行化，我们就可以充分利用好1024核的计算能力了。但想让一个程序100%并行化实在是太难了，因为你不可避免地需要在多个线程间进行同步，好让它们能彼此协作以共同完成同一个计算任务。就好比雇用1024个作家来同时创作一个剧本，需要召集大家开会，进行同步以协调彼此的工作，而1024个作家彼此沟通、同步的成本之高，很可能让整个剧本创作时间剧增，最终远达不到加速1024倍的目标。

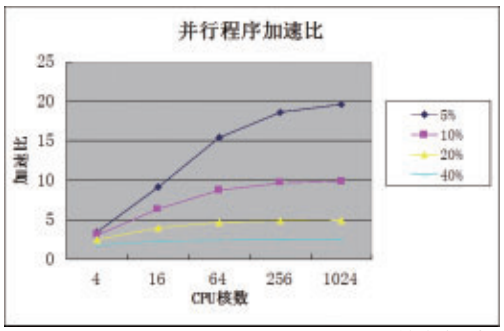


图1 并行程序加速比

让我们来看一个简单的例子。图1中描述了并行程序在4核到1024核处理器上的加速比。4条不同的曲线分别代表串行部分所占比例分别为5%、10%、20%和40%时的加速比。从图中我们可以看到，在串行部分只占5%的情况下，1024核CPU上该程序仍然只能达到20倍左右的加速比。让人惊讶的是，如果串行部分占到了40%，该程序在1024核上最多只能被加速2.5倍。

由此可见，“同步”所带来的串行化开销是并行程序最大的性能杀手，如果这个问题得不到有效解决，在1024核处理器真的到来的那一天，恐怕我们也没办法利用好它的计算能力。如果这个性能问题得不到解决，它带来的影响也会是灾难性的，因为这意味着软件再也不能有效利用硬件性能的提升了，IT产业发展的基石将会动摇。

性能与正确性

为了解决同步带来的开销，许多高性能并行编程技术应运而生。无锁算法（Lock-free algorithm）是一类重要的并行算法，它通过通过原子操作（例如compare-and-swap等指令）来取代锁，以

完成相应的同步操作。然而，只要有同步，就意味着会有串行开销，也许在32核上它的加速比能接近32，但最终在千核上提升仍然会受限。Read-Copy-Update（简称RCU）是另一种低开销的并行算法，与无锁编程类似，本质上RCU还是通过较低开销的原子操作来实现同步。

函数式编程是另一种备受推崇的并行编程范式，其关键原因在于函数式编程语言严格禁止side effect，即“副作用”。简单来说，函数式语言规定类似f(x)这样的函数，其行为只能根据x这个输入，产生一个相应的返回结果，除此之外，f(x)不能对任何其他状态或外部环境产生影响。

例如下面的这段代码中，func1()和func2()彼此不会相互干扰，func1()一定不会干扰到r2的值，因此它们之间的执行顺序不存在依赖关系，可以并行执行。而一般的命令式编程语言中，由于r1的计算过程可能修改某些系统环境变量（例如某个全局变量），而r2在计算时又依赖这个变量，从而导致它们不能被并行执行。

```
var r1 = func1();
var r2 = func2();
var r3 = func3 (s1, s2);
```

然而，并不是说函数式编程就完全不需要依赖关系了。例如求解f(g(x))，它隐式地规定了必须先求解g(x)才能再去执行f()这个函数。这样的顺序依赖关系仍然会造成串行，从而影响加速比。

我们为什么需要同步？如果不同步的话会出现什么问题？回到之前1024个作家共同写作的例子，如果他们彼此不用做任何沟通，而是只负责各自章节的写作，那么他们也许可以只花1/1024的时间就能完成整个剧本。但这种没有沟通的剧本，剧情之间很可能毫无联系，上下文的衔接也会漏洞百出。归根结底，如果不同步，我们虽然能得到性能的提升，却丧失了“正确性”。

当然，并不是所有的应用都需要同步。让我们看看自然界中一些神奇的并行现象吧。图2分别描述了3种不同的群体动物，第一个是鸟群，第二个是鱼群，第三个是蚁群。非常神奇的是，这三类动物彼此之间并没有进行“同步”，但它们却共同形成了一个让人惊叹的群体行为。以鱼群为例，它们之间的行为具备两类基本特性。

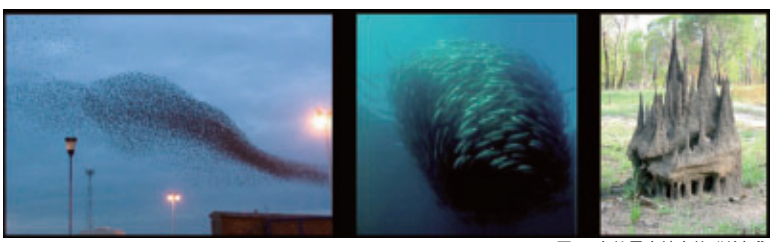


图2　自然界中神奇的“并行”

- 彼此分离。鱼与鱼之间保持固定的距离，这样就不会出现相撞。
- 整齐划一。很神奇的是，鱼群会按照某种固定形状结队，例如形成旋涡使得其天敌无从下嘴。

在感叹自然界的鬼斧神工的同时，我们不禁会问，这样的“并行”，能否给我们一些新的启发？在进行并行计算时，我们期望不同的线程最终能把它们各自的结果汇总到一起，从而形成一份最终的结果。具体到作家的例子，我们希望1024位作家能把他们各自负责的章节最终汇总到一起，形成完整连贯的剧本。可是，如果1024个章节本身就不存在关联，那不就不需要同步了吗？例如，我们只是希望写1024份不同的剧本，那么我们自然可以请所有的作家同时写作，各干各的，最终产出一个由1024份剧本构成的“组合”。在前几期的《并行编程的曙光》和《云计算时代的多核开发》中，我向大家介绍过可以直接并行化的应用：如果多核上运行的任务是针对不同用户的，彼此独立的，那么就可以非常自然地用好多核，而不需要为同步带来的开销担心了。这种彼此独立的玩法，就是在千核CPU上进行编程的一种解决方案。

转变思路：结果差不多对就行

为了追求性能，我们是否能在正确性上做一些妥协呢？如果大部分结果都是正确的，只有小部分“错误”需要修正，是否能让人接受呢？在2011年11月的SPLAH学术大会上，IBM研究院的David Ungar博士做了名为《Everything You Know about Parallel Programming is Wrong! – A Wild Screed about the Future》的主题演讲，向大家介绍了一种新的并行编程思路。在演讲中，他在“性能”与“正确性”之间做了一个出乎意料的选择：为了追求性能，我们为什么不可以放弃一部分正确性？

放弃正确性？这个想法是不是太疯狂了？人类的自我认知中最重要的环节之一就是对确定性的渴望：难道你希望有一天多核计算机在计算1+1时，得到的结果不是2吗？这简直是反人类啊！

David在SPLASH大会的演讲中给出了如下例子。图3中由上至下按时间顺序描绘了这样一个过程：Alice在1A步时提出请求，希望得到一个商品的售价，该计算从2A步开始到3A步时执行完毕。在2A和3A步之间，Bob将商品的单价改为10元，并在2B步时通知这个商品的价格缓存的结果是错误的。这样的话，在3A步时计算并缓存的结果就是错误的，因为Bob更新了商品价格。同样的，Cathy在1C步提出请求时读到的结果也是错误的结果。终于，在第1F步时，名为Freshener的线程根据更新后的商品价格重新计算了该值，并将结果缓存。这样，在此之后的1D步和1E步从缓存中读到的两个值就都是正确值了。



图3　异步并行修正方案

可以看到，这个协议非常简单，但它却代表了一种新的思路：让程序并行地去执行，如果某次运算结果被发现是错误的话，就派个代表（即Freshener）去重新计算。

这个思路给我们的启示是，如果只有极少部分的查询会返回错误的结果，并且错误率可以接受，我们就可以在兼顾正确性的同时极大提升千核上的并行性能！虽然这个想法解释起来很简单，却是非常重要的一个思维转变。要知道，很多重要的创新往往是从一个思想的根本性转变开始。如果我们一味追求正确性，所以只能想办法降低同步开销，这没错，但这条路最终只会越来越窄。哪怕同步开销降再低，低到在千核上我们仍能得到高加速比，但万核的时候怎么办？万万核呢？你又该怎么玩？反过来，如果能跳出思维惯性，从另一个角度思考问题，可能创新的机会就随之而来了。且不说David提出的这个思路最终是否能走得通，但在读到他这个看似疯狂的想法时，我确实被震撼了！正所谓不破不立。

David提出的这套机制最终能有多好的性能和准确率呢？他在16核的机器上分别使用0–14个Freshener进行了测试，结果如图4所示。如果每个工作线程配备一个Freshener，最终有小于20%的结果是过期的错误；其中90%的过期错误查询少于8次。总体而言，只有2%的结果是不可接受的。这个测试结果表明，如果容错机制足够好，我们确实有可能在满足正确率的前提下实现真正高性能、不需要同步的并行编程。当然，这个研究方向还只是起步，很多问题有待解决。

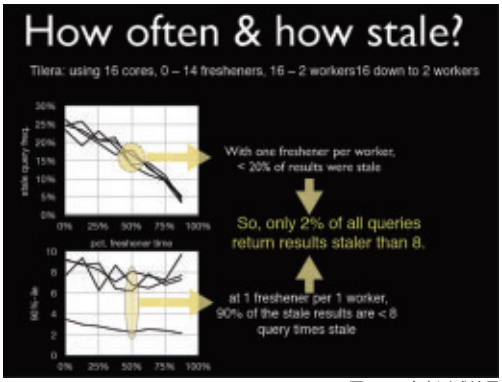


图4　正确度测试结果

总结

因为许多的应用需要进行“同步”才能得到正确的结果，在未来的千核平台上并行地运行这些应用面临着重大的性能挑战。David Ungar博士提出了一种新思路，通过牺牲部分正确性来避免同步，从而提升并行程序在多核处理器上的性能，希望这个思路对大家有所启发。



陈冠诚

IBM中国研究院研究员，主要研究方向为并行计算、计算机体系结构及大规模分布式系统。个人博客为http://parallellabs.com。

责任编辑：卢鹄翔（ludx@csdn.net）

新书上架

本月新书



深入浅出CoffeeScript

作者：Trevor Burnham

译者：寸志

本书通过一个贯穿全书的小游戏，从基础知识讲起，全面透彻地介绍了CoffeeScript，并展示了它与jQuery以及Node.js如何搭配运行。



HTML5移动Web开发指南

作者：唐俊开

本书讲述如何利用HTML5相关技术开发移动Web网站和Web App应用程序。同时涵盖了JavaScript移动开发框架jQuery Mobile、Sencha Touch及PhoneGap。



瓷片的诱惑：Windows Phone应用程序开发快速入门

作者：屠建飞

本书内容详尽，涵盖了Mango系统的各项特性与应用开发技术；实例丰富，多达百余个，且贴近应用开发实战，可用作实际开发参考。



软件测试的艺术（原书第3版）

作者：Glenford J. Myers　Tom Badgett　Corey Sandler

译者：张晓明　黄琳

以一次自评价测试开篇，探讨了代码检查、走查与评审、测试用例的设计、模块（单元）测试、系统测试、调试等主题，以及极限测试、互联网应用测试等高级主题，全面展现了作者的软件测试思想。



无懈可击的Web设计：使用HTML5和CSS3提高网站的灵活性与适应性（第3版）

作者：Dan Cederholm

译者：马跃

本书结合实例讲述了如何采用标准设计策略来满足以各种方式浏览网页的各类用户的需要。

推荐图书



JavaScript权威指南（原书第6版）

作者：David Flanagan

译者：淘宝前端团队

出版社：机械工业出版社

《JavaScript: The Definitive Guide: Activate Your Web Pages》的第6版是2011年4月推出的。当时我在O'Reilly官方网上商店购买《The Book of CSS3》，然后就顺手多买了一本最新版的《JavaScript: The Definitive Guide: Activate Your Web Pages》。

在过去的半年里，我分别给Adobe、Canon和CA做了几轮JavaScript和HTML5的培训。每次做完培训后，都会有人问我如果想要继续深入学习该看什么书好，我都会说，如果你能够把《JavaScript: The Definitive Guide: Activate Your Web Pages》看完，那就很了不起了。

对于JavaScript新手来说，其实看什么书不重要，重要的是要能引起你兴趣使得你愿意动手做项目。在动手做过几个项目，对了JavaScript有了感性认识之后，你可能就会想要学习一些进阶内容。这时，《JavaScript: The Definitive Guide: Activate Your Web Pages》就会成为最佳的选择。为什么在JavaScript里面可以使用objA||objB以及obj&&obj.prop的写法？为什么在使用for ... in时toString不会出现，但自己附加到Object.prototype上的方法则会污染对象？这些问题入门书是不可能细说的，因为它们的任务是帮你尽快上手。这些问题可能你已听别人提到过，可能你也在自己的项目中实践过，但最根本的“为什么”却从来没有深究过。读一读《JavaScript: The Definitive Guide: Activate Your Web Pages》，你会发现平常死记硬背的概念背后其实都有理论支撑的。

对于已经写了多年JavaScript代码，并且对上述概念已经十分清晰的人来说，《JavaScript: The Definitive Guide: Activate Your Web Pages》唯一有价值的地方就在于“第 6 版”这 3 个字上了。这一版的主要更新在于涵盖 ECMAScript 5和HTML5的内容。对于ECMAScript 5，你可以去看ECMA 262，里面每一条都说得十分清楚，不存在任何歧义，但也不保证你理解后就知道怎么用。这就好比说，给你一把全新的电动工具，附带详细的说明书，但你看完之后还是不明白下次要在家里DIY点东西时如何把它用上。类似的，你看完ECMA 262里对Array.prototype.reduce操作步骤的详细说明，可能意识不到可以用它来写一个sum函数，也不知道这个sum函数该如何写。《JavaScript: The Definitive Guide: Activate Your Web Pages》的价值就在于，给予你工具的同时也通过示例告诉你什么情况下应该如何使用工具。

你知道用到了Drag&Drop、Blob还有File API，然而却发现除了拼凑来的代码以外，并不知道这3项技术分别还能做什么，以及在什么别的场合下应该想到要使用它们。这时，你就应该阅读《JavaScript: The Definitive Guide: Activate Your Web Pages》了。

说完为什么我推荐看这本书，那么接下来我们讨论一下到底应该买英文版还是中文版。如果你想要锻炼自己的英文阅读能力，又或者你想要在iPad或Kindle上随时随地地阅读，那么可以去买英文版。如果希望尽快流畅地读完整本书，又或者觉得电脑桌上用来垫笔记本电脑的那本《代码大全》该换掉了，那么你就买中文版吧。

点评人：陈广琛，网名Cat Chen，豌豆荚前端架构师，曾任百度前端研发部高级工程师。



苹果狂潮：iPhone开启永远在线的时代
作者：Brian X. Chen
译者：苏健
出版社：人民邮电出版社

用户体验至上，是苹果一贯的作风，而且苹果比别人都清楚，开发者也是用户。xcode除了容易崩溃外，几近是用户体验最好的IDE。虽然苹果认为第三方开发团队才是最了解市场需求的草根，却缺乏售卖软件的营销渠道和支付渠道，还同时倍受盗版打击。于是App Store横空出世，为众开发者提供了广阔的商机，同时也为苹果招募了大量的开发者。这些开发者（也许也有你）为苹果和苹果用户创造了数十万五花八门的应用，用户只需轻点几下手指，就能安装到iPhone中。

其实，苹果比我们看到的可能还要宽广——从iPhone的软件系统、硬件的工业设计、电子电路设计和生产、让iPhone开发者又爱又恨的App Store，再到整个新媒体革命。《苹果狂潮》这本书的前半部，将会带你从一个开发设计者的角度站出来，从更高的层面去看待整个苹果的垂直产业链。

《苹果狂潮》中提到，微软CEO鲍尔默在2007年1月接受CNBC采访时，对iPhone评论道："这是世界上最昂贵的手机，而且由于它没有键盘，故而不是一台很好的电子邮件收发机，因此并不能吸引商务用户。"然而在5年后的今天，手机邮件甚至短信、电话都变成了附属功能，手机正在全面替代传统的媒体和娱乐方式。比如，用手机刷微博、玩游戏或把玩一下其他手机应用。相信鲍尔默当时没有预料到的是，现在手机已经浸入了我们的生活，并成为生活的重要组成部分。而在手机从"任何事情、任何时间、任何地点"多方面浸入我们生活的同时，一些现象也慢慢浮现出来——人们在"任何事情、任何时间、任何地点"都已完全无法离开互联网。很多人担心对互联网的依赖最终将让人类的思考变得"浅薄"，并对新形式的娱乐上瘾。大家担心，像Facebook、微博这类型的社交工具，会抢占人们社交的时间，让人们逃避现实，变得孤僻、内向。

科技革命给我们带来了什么？这也是《苹果狂潮》作者Brian X. Chen一直以来在思索的问题。书中说到艾比林基督大学对Facebook进行研究，发现它实际上是一面镜子，反射出了现实生活中参与者之间的互动。那些Facebook上活跃地与人们联系的学生在现实世界中本来就是热衷社交的人。因此，社交工具应是对线下活动的延伸，是网络让社交变得更方便。而整个互联网对人们的改变，也将在未来被印证为积极的变革。这就像是活字印刷之前，书籍都是由称作"文士"的抄写员做成手抄本。然而在15世纪末，活字印刷开始兴起时，部分宗教人士大为恐慌，他们认为这破坏了人们的虔诚，本应由文士亲自抄录，现在却是冰冷的机器。1565年，瑞士科学家康拉德对印刷书籍也进行了批评，他宣称信息过载将会让现代社会不堪重负。然而，现在恐怕手抄的只有作业了吧。

"任何事情、任何时间、任何地点"，永远在线的时代已经开启。《苹果狂潮》就是要带领各位，一起探寻永远在线社会的意义。

点评人：包季真，《触动人心：设计优秀的iPhone应用》译者，《脉可寻》产品总监 。

全球排行榜

Amazon

01 HTML and CSS: Design and Build Websites
02 Cracking the Coding Interview: 150 Programming Questions and Solutions
03 JavaScript & jQuery: The Missing Manual
04 iOS Programming: The Big Nerd Ranch Guide，3rd Edition
05 Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Edition
06 In The Plex: How Google Thinks, Works, and Shapes Our Lives
07 The Shallows: What the Internet Is Doing to Our Brains
08 Turing's Cathedral: The Origins of the Digital Universe
09 Game of Thrones: Prima Official Game Guide
10 slide:ology: The Art and Science of Creating Great Presentations

天珑书局（中国台湾）

01 App程序设计入门：iPhone、iPad（第2版）
02 iOS创意程序设计家：iPhone+iPad跨平台通用
03 《超强图解》前进Android Market！Google Android SDK实战演练（第2版）
04 iOS SDK开发范例大全
05 JavaScript设计模式
06 Android系统原理深入解析
07 打造安全无虞的Web Applications
08 HTML5·CSS3精致范例辞典
09 精通Objective-C程序设计（第4版）
10 版本控制使用Git

第二书店

01 WCF全面解析（全2册）
02 七周七语言：理解多种编程范型
03 白帽子讲Web安全
04 Java编程思想（第4版）
05 算法导论（原书第2版）
06 PHP和MySQL Web开发（原书第4版）
07 Java核心技术，卷1（原书第8版）
08 Linux/Unix设计思想
09 Node Web开发
10 SEO艺术

GEEK产品



全景球形相机

柏林工业大学的研究人员研发了一款360度全景球形相机，它的上面集合了36个200万像素的镜头。将它抛向天空后，球飞至顶点时，所有镜头就可以同时打开，拍下一张360度全景照。



BeoPlay A3

B&O近日推出一款iPad便携音响底座BeoPlay A3，可以完全将iPad嵌入其中，同时可以调整成四种立面角度来看，可支持8个小时的外放时间。

Level 10 M鼠标

宝马美国旗下设计部门BWM Designworks与TT合作，推出了这款Level 10 M鼠标。
它采用几乎全开放式的设计，允许用户通过旋转螺丝自行调整鼠标上部的高度、角度和整体大小。在手心位置给容易出汗的用户预留了散热孔，人性化的设计使得夏天长时间使用也不会感到不适。





iBGStar血糖测试仪

这款血糖测试仪可以通过一个30Pin的接口连接到iPhone以及iPod touch。和传统的血糖测试仪一样，iBGStar也采用一次性的血液采样试纸，每一个血糖测试仪都会附送50片试纸。



SpyNet Night Vision Mission Video Watch

国外设计的一款SpyNet Night Vision Mission Video Watch。
这块表最大的乐趣就是可以用它当摄像机、录音机、照相机甚至夜视镜头，即使在夜间也可以轻松使用。

“bento”计算机

美国设计师rené lee设计的“bento”（便当）概念计算机系统，将笔记本、平板电脑和智能手机融合到一个设备中去。电池和磁盘存储器可以单独升级替换，不用更换整套设备。





硅谷新传奇

Kevin Systrom

文 / 高松



一个名为Kevin Systrom的北欧裔小伙儿，用18个月创造了一款火爆的产品，并以10亿美元的价格卖给了Facebook。这一切是怎么做到的？他创造的硅谷传奇不只是在合适的时间、地点，外加一个绝妙的点子那么简单。

Kevin Systrom生于1982年，在波士顿长大，毕业于斯坦福大学，拥有管理与工程学士学位，热衷摄影。大学时他接触到最新潮的互联网创业，参加了学校组织的“Mayfield Fellows”项目，在Odeo进行暑期实习。Odeo是Twitter的前身，聚集了诸如Jack Dorsey、Biz Stone和Evan Williams等如今叱咤风云的创业明星。在他们手中，Blogger、Twitter和Square等伟大的产品脱颖而出。“在Odeo的第一天，我就决定创办自己的公司”，Systrom说道。

在斯坦福求学时，Systrom曾做过一款图片分享类应用。那时Facebook也在做类似的应用，于是希望能邀请Systrom加入，但被他拒绝。

毕业后，Systrom加入据称是最难进入的Google的收购部门，开始他的第一份工作。他总是对社交产品特别感兴趣，还参与过Gmail、Google Reader等项目。

离开Google后，Systrom到一家旅行业务创业公司Nextstop担任产品经理。非科班出身的Systrom开始在晚上自学编程，实现自己的一些产品想法，其中有一个结合了签到服务和战争游戏元素的产品，这就是此后的《Burbn》。Systrom用HTML5做出《Burbn》的原型，传给几个朋友试用，并利用周末为朋友们改进这个原型。

在一次聚会上，Systrom遇到了来自Baseline Ventures和Andreessen Horowitz的投资人，他展示的《Burbn》原型引起了投资人的兴趣。带着从Baseline Ventures和Andreessen Horowitz融到的50万美元，Systrom从Nextstop单飞，开始组建自己的团队。

《Burbn》慢慢积累了一些用户，但它的体验并不好，甚至还经常有信息延迟，于是Systrom邀请了校友Krieger兼任工程师和用户体验设计师，一起创业。他们发现用户不仅用《Burbn》签到，更倾向于用它分享照片。《Burbn》就像一个钩子，让他们找到了用户的真实需求，为此他们确定了产品的新方向：内置许多滤镜特效，加大分享功能。历经几个月的测试、修正和一款名为《Scotch》的应用失败后，Systrom和Krieger对照片分享应用的认识更加深刻，他们把这件事变得极为简化：只留下“拍照、加滤镜、分享”这三个步骤，这款新的产品就叫做《Instagram》。《Instagram》做了一些看上去特立独行的事情，例如照片默认公开、没有网站版本等，从中我们隐约能看到Twitter对它的影响。

2010年9月20日，《Instagram》正式上线，当天就有2万5千用户下载。Systrom曾表示担心用户的热情会退却，增长后继无力。可事实上，《Instagram》的用户数始终在快速增长。同年12月，iPhone 4的发布更是为《Instagram》加上了助推器。它在3个月内获得了100万用户，随后快速突破1000万，令人惊讶的是，达到这个成就完全依靠于用户的口碑相传。

此时，Facebook又向Systrom抛出橄榄枝，目标是收购《Instagram》，但再次被他拒绝。《Instagram》的成功让Systrom赢得了世界的瞩目，他获得了英国首相Cameron的接待，并多次受邀来到中国参加论坛。2011年4月，Systrom在科技音乐节SXSW上宣布《Instagram》用户数达到2700万。他说相比照片分享应用，《Instagram》更像是一个社交网络，在此，他们看到了Facebook级别的社交互动。

这无疑会令Facebook关注，于是Facebook开始第三次对Systrom的游说，这次由Mark Zuckerberg亲自出马。接下来就如本文第一句所描述的，创立仅551天的13人团队，在Systrom的带领下，创造了硅谷传奇。